Children Learning By Doing

Squeak Etoys on the OLPC XO

Viewpoints Research Institute

Viewpoints Research Institute

A non-profit public-benefit 501(c)3 organization devoted to children's education and systems research. All of our results are free and open source.

© Viewpoints Research Institute, April 2007
1209 Grand Central Ave
Glendale, California 91201

Children Learning By Doing – Etoys on the OLPC XO

Was written by Alan Kay, and is made from a variety of previous Viewpoints Research Institute writings adapted for the XO, plus new material done especially for this book.

Some of the researchers who directly contributed to and built major parts of Etoys:

Kazuhiro Abe
Bob Arning
B.J. Allen-Conn
Diego GomezDeck
Bert Freudenberg
Dan Ingalls
Alan Kay
Ted Kaehler
John Maloney
Yoshiki Ohshima
Andreas Raab
Michael Rueger
Kim Rose
Lex Spoon
Scott Wallace
Takashi Yamamiya

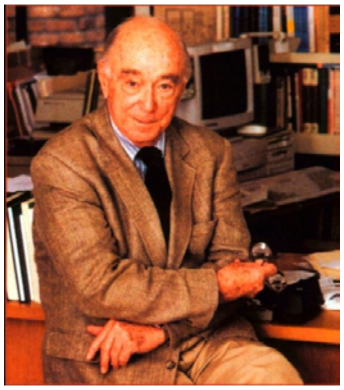
Other research colleagues, in addition to the Board of Advisors, whose ideas advanced the Etoys design:

Hal Abelson
Bill Atkinson
Bobby Blatt
Andy diSessa
Mary Laycock
Julia Nishijima
Mitchel Resnick
Julaine Salem
Brian Silverman
Cynthia Solomon
Mike Travers

This document is a first draft of a summary of Etoys on the One Laptop Per Child XO computer. It will undergo many changes in the next few months. The first official edition is expected by August 2007.

Dedications

Seymour Papert: *"Should the computer program the kid, or should the kid program the computer?"*


Jerome Bruner: *True learning is "figuring out how to use what you already know in order to go beyond what you already think."*

Preface

This document is an early draft of an introductory book about Etoys on the OLPC laptop for teachers and parents. It is planned to be completed this summer in time for XO experiments around the world.

We are putting out this unfinished draft to help provide context for the OLPC "Countries Meeting" on April 25th 2007 in Cambridge, Mass.

Though incomplete, we have attempted to leave most of the unfinished parts at the end, and provide a coherent a narrative as possible. To be added are many more examples, more discussion of curriculum and approach to math and science learning, and quite a bit more about the kinds of media authoring and collaboration that can be done in Etoys.

Table Of Contents

Dedications, Preface, Table of Contents. 3

Etoys is: 4
A Worldwide, Multilingual, Authoring and Learning Environment 5
"Like LOGO" with Costumes 6
"Like Starlogo" At All Scales. 7
"Like Hypercard", Presentations, Desktop & Web Publishing, Multimedia 8
"Like Squeak Smalltalk", Multiplatform, etc. 9

Examples: Detailed Table Of Contents 11

Educational Examples 34

Multimedia Examples 45

Squeak Smalltalk xx

Acknowledgements xx

References xx

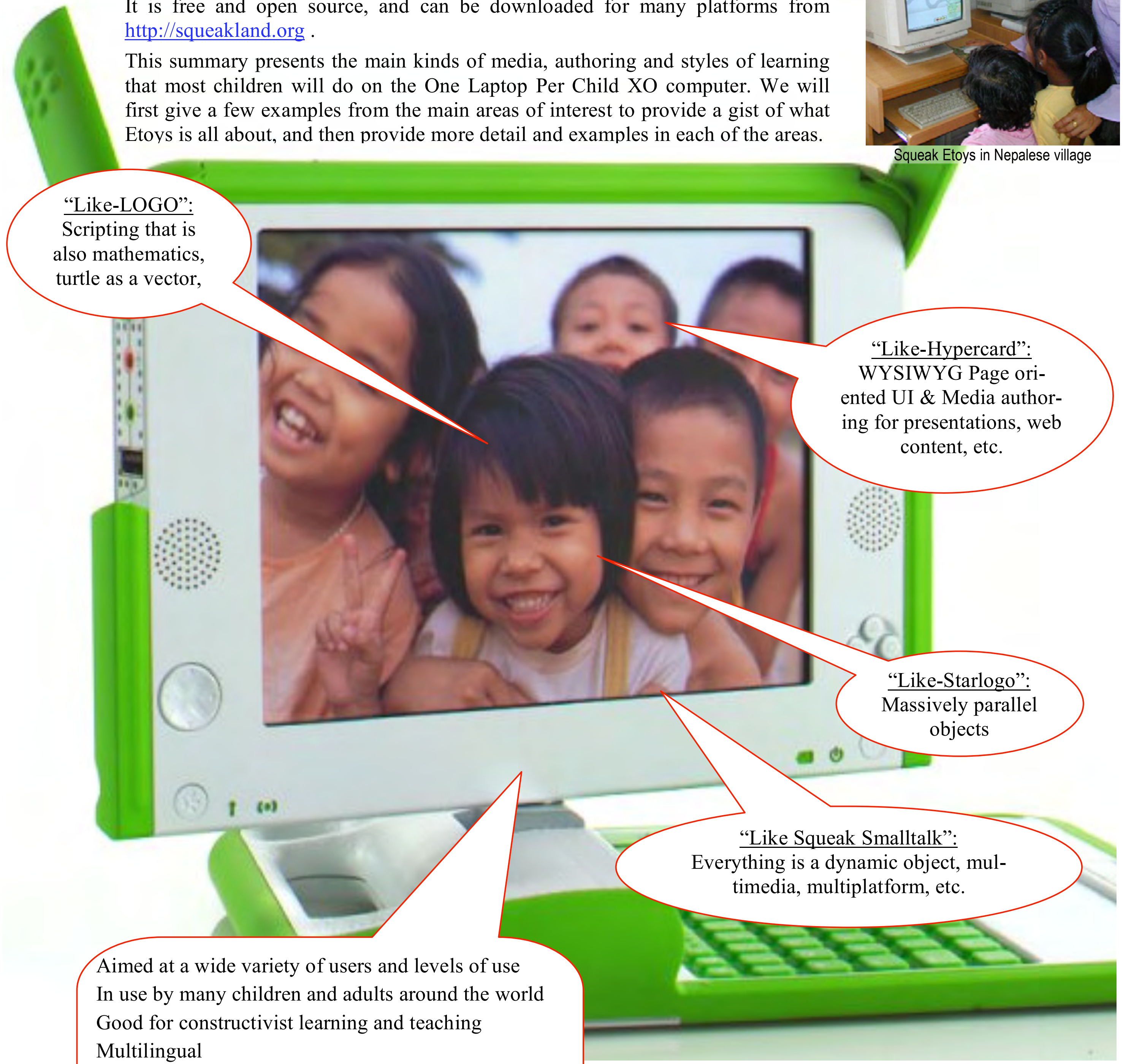
Etoys & Squeak on the OLPC XO

Squeak Etoys is a multimedia authoring system especially aimed at helping children learn powerful ideas by constructing them. It was inspired by LOGO, Smalltalk, Hypercard, and Starlogo and presents a unified style, user-interface, media and scripting environment for making things from computer stuff.

It is free and open source, and can be downloaded for many platforms from http://squeakland.org .

This summary presents the main kinds of media, authoring and styles of learning that most children will do on the One Laptop Per Child XO computer. We will first give a few examples from the main areas of interest to provide a gist of what Etoys is all about, and then provide more detail and examples in each of the areas.

Squeak Etoys in Nepalese village



"Like-LOGO": Scripting that is also mathematics, turtle as a vector,

"Like-Hypercard": WYSIWYG Page oriented UI & Media authoring for presentations, web content, etc.

"Like-Starlogo": Massively parallel objects

"Like Squeak Smalltalk": Everything is a dynamic object, multimedia, multiplatform, etc.

Aimed at a wide variety of users and levels of use
In use by many children and adults around the world
Good for constructivist learning and teaching
Multilingual
Self contained and runs on many platforms
Both standalone and web-based
Integrated multimedia
Collaborative
Made from many integrated media objects
Authorable at all levels from end-user to expert

Etoys is a Worldwide Authoring & Learning Environment

Etoys is

"Like Logo" – but with costumes, multimedia, etc.

"Like Starlogo" – but at all levels of scale

"Like Hypercard and Powerpoint" – but simpler and richer

"Like Smalltalk" – it is Squeak Smalltalk underneath

"Like itself" – it has special properties that are unique

Just "how like" will be shown on the next few pages.

Etoys was first tested with children in 1997, and has since spread around the world to be used by many children in cultural and language environments. Etoys is multilingual and has been successfully used in USA, Europe, South America, Japan, Korea, India, Nepal, and elsewhere.

The multilingualization is dynamic: languages can be switched on the fly (this can be illuminating for children) and there is a "kit" that aids the introduction of a new language.

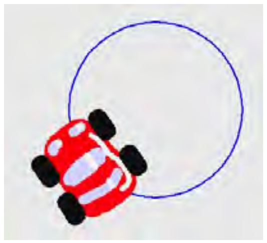
Etoys is also "ecumenical": it runs on more than 20 platforms bit-identically including all of the standard ones, many PDAs and SmartPhones, and on the OLPC XO machine.

Current Languages include:
English, French, German, Spanish, Portuguese, Japanese, Chinese, Korean, Swahili

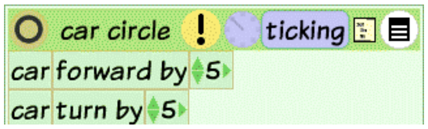
Languages in progress include:
Arabic
Greek
Thai

The main website http://www.squeakland.org contains downloads, tutorials, example content, access to other web sites, essays, and books, including "Powerful Ideas in the Classroom", an introduction to Etoys for 4-5-6 grade teachers.

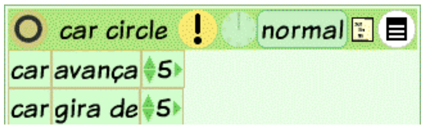
There are two international conferences each year that are centered on Etoys and similar environments: IEEE C5 conference (usually in Jan) and SqueakFest (usually in Aug).



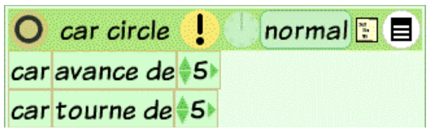
Below are Logo-like scripts in different languages to move the car in a circle .



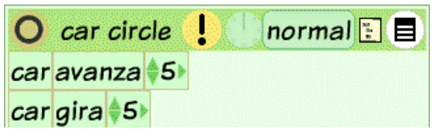
```
car circle ! ticking
car forward by 5
car turn by 5
```



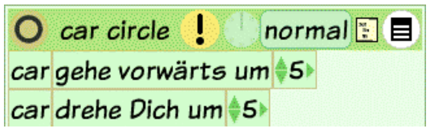
```
car circle ! normal
car avança 5
car gira de 5
```



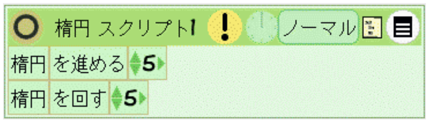
```
car circle ! normal
car avance de 5
car tourne de 5
```



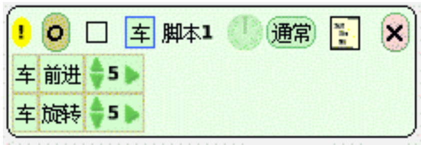
```
car circle ! normal
car avanza 5
car gira 5
```



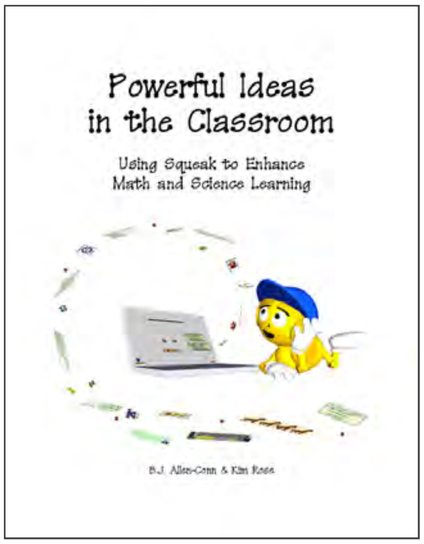
```
car circle ! normal
car gehe vorwärts um 5
car drehe Dich um 5
```

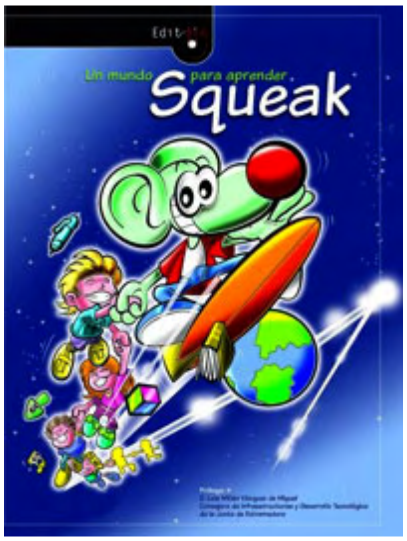


```
楕円 スクリプト1 ! ノーマル
楕円 を進める 5
楕円 を回す 5
```



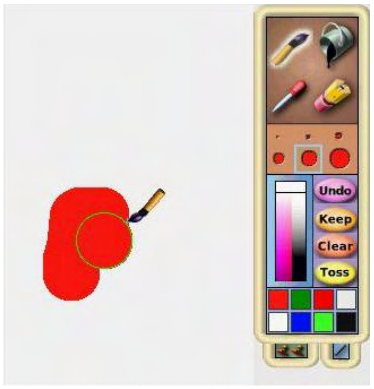
```
! 车 脚本1 通常
车 前进 5
车 旋转 5
```

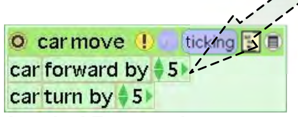

Powerful Ideas in the Classroom
Using Squeak to Enhance Math and Science Learning
B.J. Allen-Conn & Kim Rose



Etoys is "Like LOGO": But With Universal Objects & Costumes

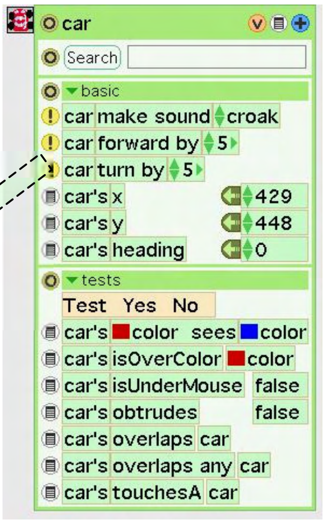
A project that nine, ten and eleven year old children all over the world love is to design and make a car they would like to learn how to drive. They first draw their car (and often put big off-road tires on them like this).





```
carmove ! ticking
car forward by 5
car turn by 5
```

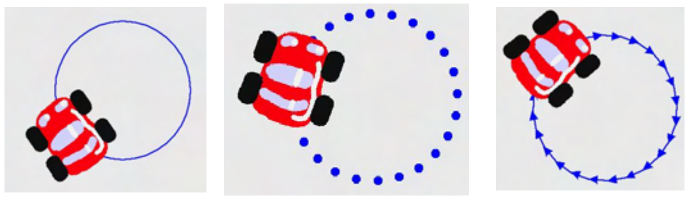
A typical Etoys script



```
car
Search
basic
car make sound croak
car forward by 5
car turn by 5
car's x        429
car's y        448
car's heading  0
tests
Test  Yes  No
car's 🟥color sees 🟦color
car's isOverColor 🟥color
car's isUnderMouse  false
car's obtrudes      false
car's overlaps car
car's overlaps any car
car's touchesA car
```

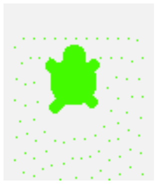
Etoys "viewer" for the car

So far this is just a picture. But then they can look "inside" their drawing to see its properties (for example where the car is located and heading) and behaviors (the ability to go forward in the direction it is heading, or change its heading by turning. These behaviors can be pulled out and dropped on the "world" to make a script – *without the need for typing* – which can be set "ticking" by clicking on the clock. The car starts moving in accordance with the script.

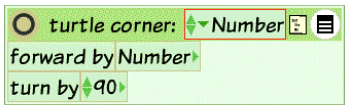


If we drop the car's pen on the world, it will leave a track (in this case a circle), and we see that this is Papert's LOGO turtle in disguise – a turtle with a "costume" and easy ways to view, script and control it.

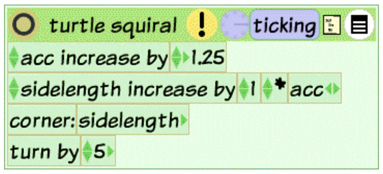
In fact, it is easy to make a Logo in Etoys, and it can take advantage of the "no keyboard input" feature. We can just draw a turtle, as we did with the car.



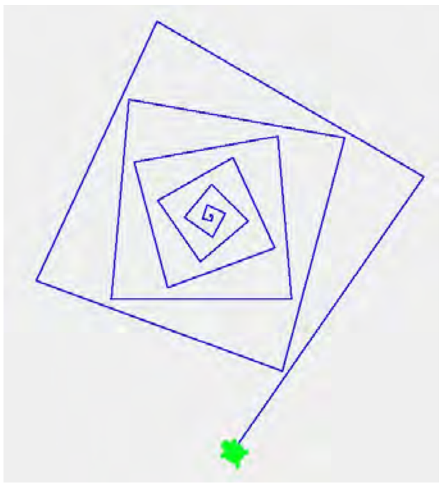
In Etoys we can choose or now to show the object's name in a script (to not show it is classical Logo style). So we can make a script that will draw a line of some length and turn a corner. Then we can invoke this script to make many kinds of geometric figures:



```
turtle corner: Number
forward by Number
turn by 90
```



```
turtle squiral ! ticking
acc increase by 1.25
sidelength increase by 1 * acc
corner: sidelength
turn by 5
```



Etoys is "Like Starlogo": But At All Scales & With Etoys Scripting

Squeak Etoys allows an enormous number of objects to be run simultaneously (inspired by Starlogo) that are scripted using the same conventions used with the larger media objects. This allows children to think through complex parts of a project – such as an ant or salmon following a scent gradient – in the large with a few large ants or fish, and then to use what has been discovered with a population of thousands of particle animals.

From scripting one object, we can make lots of them ...

The Beauty and Importance of Complex Systems

Lots of interesting things come in bunches which exhibit emergent group behavior, such as:

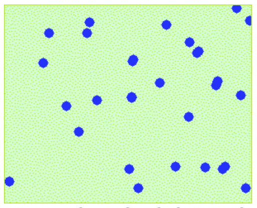
Electrons (current flow, etc.)

Atoms and molecules (gases, liquids, solids)

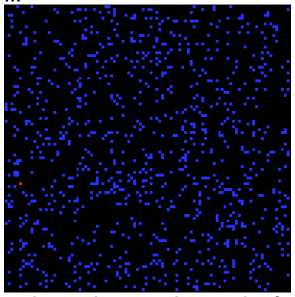
Cells

Populations of animals and plants

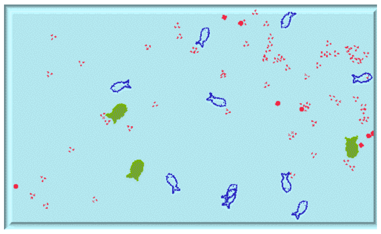
Etoys can handle many 10's of 1000s of objects at many frames per second, and this is enough to allow excellent models of the important properties of these systems to be made.

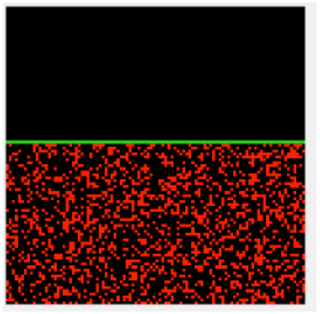

... to explore simple interactions like collisions, disease, etc., and ...

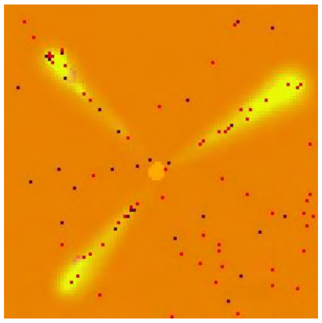
The continuity of scripting between large and small allows children to explore very complex systems by just scripting the behavior of one item and making many copies. For example, if we make lots of little dots, we can explore the behavior of contagious processes, such as rumors and disease. Here the scripts are very simple, and cause a dot to change color when it collides with an "infected" dot. The size of the arena for collisions determines the delays between collisions, and allows us to explore matters of life and death, such as really understanding the characteristics of epidemics: fast deadly ones like typhoid which are very noticeable, and slow deadly ones such as AIDS (which is deadly in part because the onset of an AIDS epidemic is not dramatic). A poor understanding of slow deadly epidemics in many parts of the world is one of the main causes of the AIDS disaster.

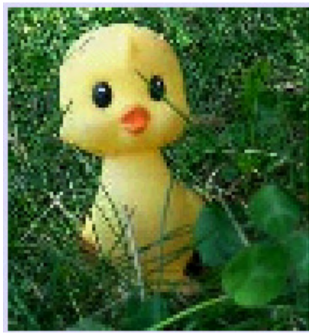

... then make many thousands of them to explore more complex systems from gases to epidemics

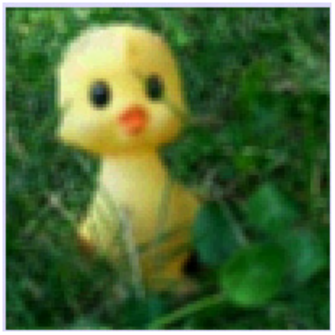
People have to reach beyond their common sense into the "uncommon sense" of models for disasters in order to help their imagination spur them to early action. Because of this new way to "understand and argue", true "computer literacy" (which involves being able to read and write dynamic simulations of ideas, especially in terms of systems of related entities, themselves also portrayed as systems) will eventually create an even larger change in how humans think about ideas than the printing press and classical mathematics and science.

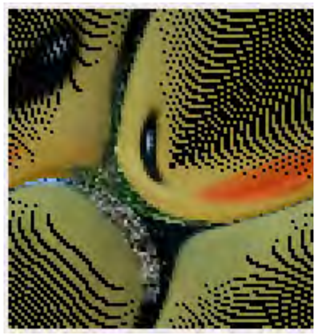

Simple Aquarium Ecology; female and male fish, and plankton


2000 "gas particles" supporting a moveable piston


... to making a simulation of ants gathering food,and leaving a diffusing trail of pheromones to guide other ants.


Here each of the 10,000 pixels in this picture is like an "ant", a particle that can be scripted. We can get


... them to do picture processing (in this case, a blur filter), but can also be handled in more interesting ways ...


Here each pixel is told to move forward a distance that is 5% its own location relative to the center.

Etoys is "Like Hypercard": But With Presentations, Desktop & Web Publishing, etc.

Media authoring style is hypercard-like and WYSIWYG (without modes)

Bring together graphical objects and organize them on a page

All objects can be scripted (using same scripting as children use).

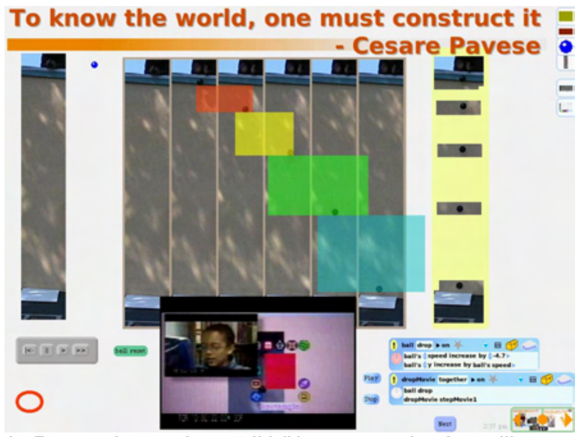
All creations can be saved, published to the web, etc.

No need for special presentation media (such as ppt)

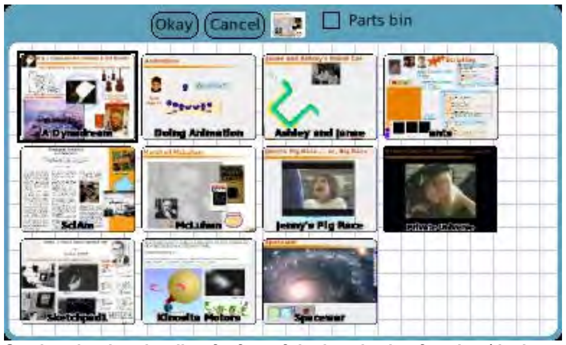
All authoring is alive on the web even in a browser

Real-time collaboration with VOIP and screen-sharing

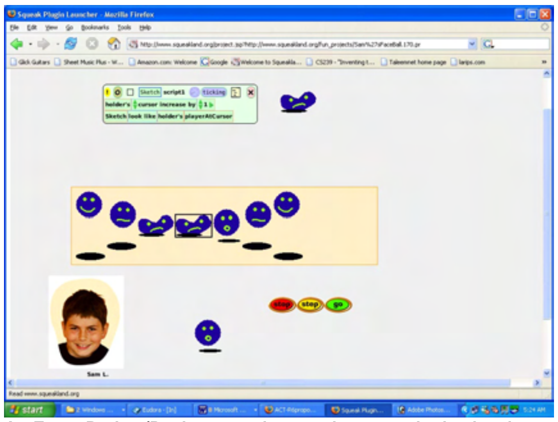
More details later on in this document (pages pp-qq).



An Etoys project used as a "slide" in a presentation, but unlike Powerpoint, all the objects and authoring are "on" and can be dynamically invoked and changed.



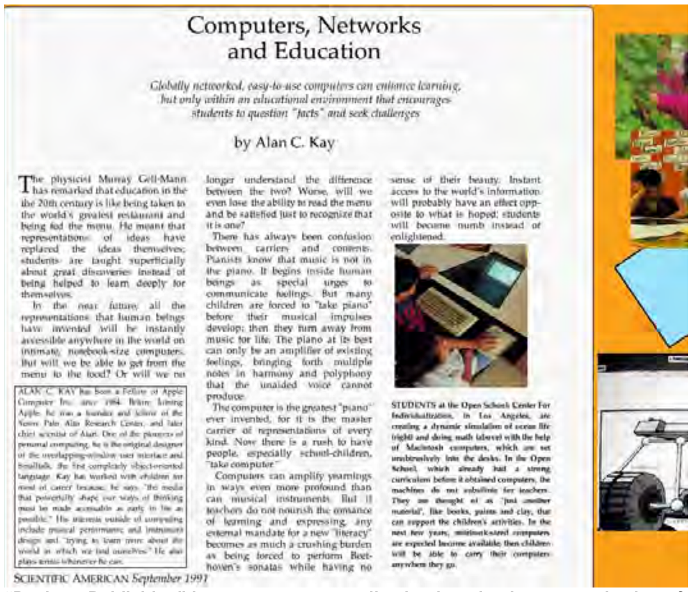
Sorting the thumbnails of a few of the hundreds of project/desktops made by this user



An Etoys Project/Desktop running as a browser plugin showing Sam's "FaceBall" animation. Full WYSIWYG authoring is always available.

Movie player can play MPEG and JPEG Movies

MP3 and other sound format player



Computers, Networks and Education

Globally networked, easy-to-use computers can enhance learning, but only within an educational environment that encourages students to question "facts" and seek challenges

by Alan C. Kay

The physicist Murray Gell-Mann has remarked that education in the the 20th century is like being taken to the world's greatest restaurant and being fed the menu. He meant that representations of ideas have replaced the ideas themselves; students are taught superficially about great discoveries instead of being helped to learn deeply for themselves.

In the near future, all the representations that human beings have invented will be instantly accessible anywhere in the world on intimate, notebook-size computers. But will we be able to get from the menu to the food? Or will we no longer understand the difference between the two? Worse, will we even lose the ability to read the menu and be satisfied just to recognize that it is one?

There has always been confusion between carriers and contents. Pianists know that music is not in the piano. It begins inside human beings as special urges to communicate feelings. But many children are forced to "take piano" before their musical impulses develop; then they turn away from music for life. The piano at its best can only be an amplifier of existing feelings, bringing forth multiple notes in harmony and polyphony that the unaided voice cannot produce.

The computer is the greatest "piano" ever invented, for it is the master carrier of representations of every kind. Now there is a rush to have people, especially school-children, "take computer."

Computers can amplify yearnings in ways even more profound than can musical instruments. But if teachers do not nourish the romance of learning and expressing, any external mandate for a new "literacy" becomes as much a crushing burden as being forced to perform Beethoven's sonatas while having no sense of their beauty. Instant access to the world's information will probably have an effect opposite to what is hoped: students will become numb instead of enlightened.

ALAN C. KAY has been a Fellow of Apple Computer Inc. since 1984. Before joining Apple, he was a founder and fellow of the Xerox Palo Alto Research Center, and later chief scientist of Atari. One of the pioneers of personal computing, he is the original designer of the overlapping-window user interface and Smalltalk, the first completely object-oriented language. Kay has worked with children for most of career because, he says, "the media that powerfully shape our ways of thinking must be made accessible as early in life as possible." His interests outside of computing include musical performance and instrument design and "trying to learn more about the world in which we find ourselves." He also plays tennis whenever he can.

STUDENTS at the Open School Center For Individualization, in Los Angeles, are creating a dynamic simulation of ocean life (right) and doing math (above) with the help of Macintosh computers, which are set unobtrusively into the desks. In the Open School, which already had a strong curriculum before it obtained computers, the machines do not substitute for teachers. They are thought of as "just another material", like books, paints and clay, that can support the children's activities. In the next few years, notebook-sized computers are expected become available, then children will be able to carry their computers anywhere they go.

SCIENTIFIC AMERICAN *September 1991*

"Desktop Publishing" is not a separate application but simply an organization of desired objects to look like a high-quality printed page. Any old or new objects can be simply dragged to become part of this "document". Here we see an article written for Scientific American by the author being organized, with possible images for the article on the right

.

Etoys is "Like Squeak Smalltalk": In Fact, It Is

In fact, underneath the Etoys user interface is Squeak Smalltalk, a full featured comprehensive all-levels programming environment whose earlier versions at Xerox PARC in the 70s included the inventions of dynamic object-oriented programming and graphical user interfaces.

This means that the OLPC has a programming and media environment roughly equivalent to Java, but much smaller, simpler and self-contained.

Squeak is free and open source.

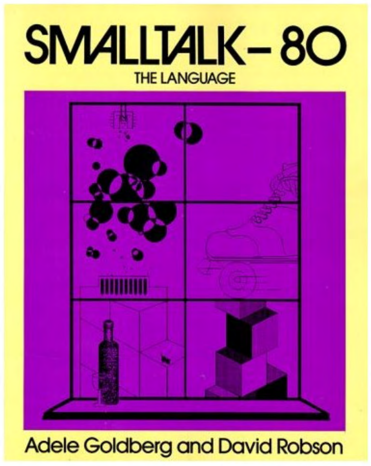
Squeak itself is a complete computer system from the end-users and user interface to the hardware of the underlying machine, and each level of the system can be dynamically examined and authored. For most young users and their teachers this is not an important consideration because they will only be using the Etoys environment level. But, especially for high school and college age students, the ability to dynamically examine and experiment with all levels of the system allows Squeak to serve as a "living laboratory" of computer science and software engineering, including: operating system principles, graphics and sound synthesis, user-interface and media design, etc. Though quite complete, Squeak is very small: only about 200,000 lines of code for everything, including its own "operating system", graphics and sound, internet sockets, file system, user-interface, applications, and Etoys itself. The lower levels are written in the high level object oriented language and the system is set up to transport itself to run "bit-identically" on new platforms in a day or so.

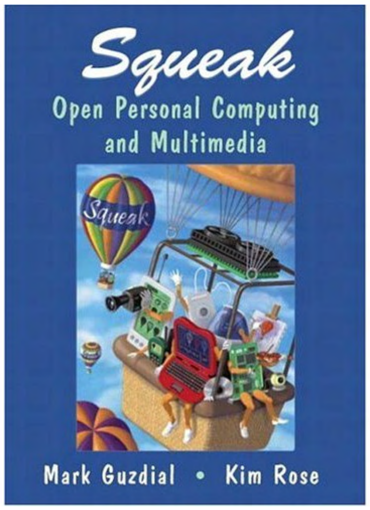
There are a number of open source communities built around Smalltalk, and several around Squeak. http://www.squeak.org is the main site for deep systems work in Squeak.

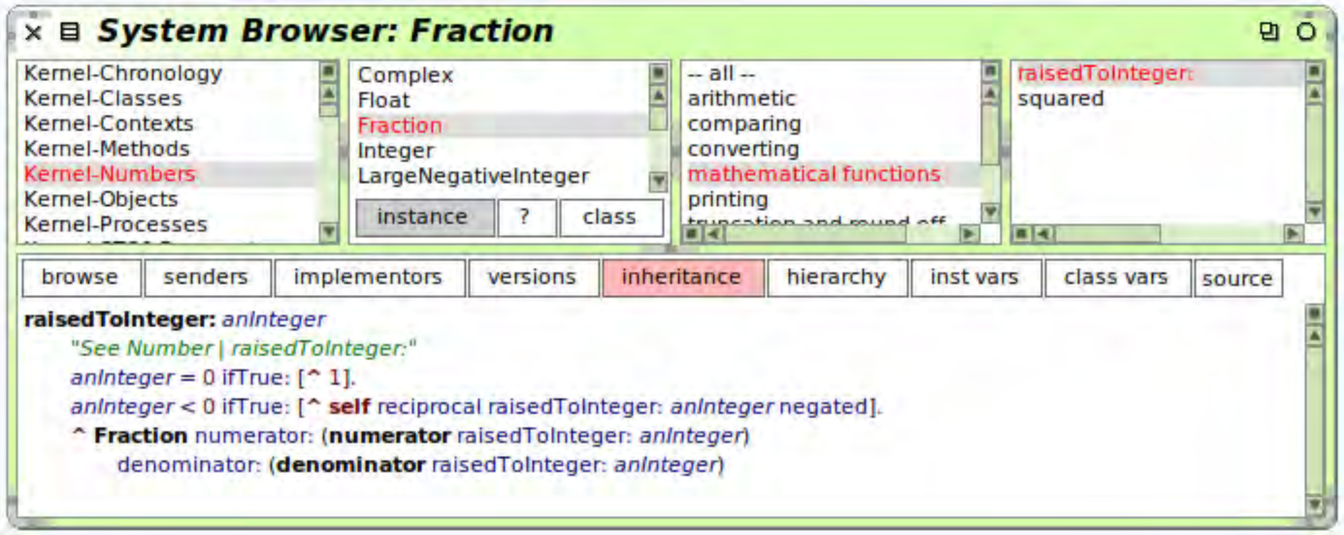
An extensive new collaborative 3D environment called Croquet can be found at http://www.opencroquet.org .

The educational work with children and much work with children can be found at http://www.squeakland.org .

There are also sites for server-side uses of Squeak, including for wikis, development for browser client-side apps, etc.







```
raisedToInteger: anInteger
    "See Number | raisedToInteger:"
    anInteger = 0 ifTrue: [^ 1].
    anInteger < 0 ifTrue: [^ self reciprocal raisedToInteger: anInteger negated].
    ^ Fraction numerator: (numerator raisedToInteger: anInteger)
        denominator: (denominator raisedToInteger: anInteger)
```

A Squeak Smalltalk systems browser showing how mixed mode arithmentic is defined in Smalltalk. Here we see the method that computes how a fraction (rational number) can be raised to an integer power. Smalltalk is completely defined in itself and the entire system is "alive" and available at all times for perusal, changes, experiments, adding new features, or to do major subsystems.

Examples Table Of Contents

Many examples are yet to be included, but enough are here to provide a gist of how we use Etoys to help children learn mathematics and science, and to make many kinds of computer constructions using the media tools in Etoys.

<u>**Educational Examples**</u>

Math and Science

Primer: Forward and turn
- Variables
- Random
- Feedback (follow path)
- Simple Animation

Increase By
Speed & Acceleration
Increase-By for vectors

Particles: Diffusion

Animation: Animations, Movies
Sound Synthesis

Costumes: Logic

Measuring Bike Wheel

Gravity: Galilean Gravity
Lunar Lander
Gas and Piston
Getting To Algebra

Inertia
Off the Cliff
Shoot the Alien
SpaceWar: Inertia and vectors
Orbits:
Springs

Particles: Gravity
Piston and gas
Vacuum Cleaner
Buoyancy, Brownian Motion

Feedback: Follow the path
Salmon
Particles: Ants

Ecologies:
Epidemic
Fish and plankton (other predator/prey)

Variation & Test: Weasel

<u>**Media Examples**</u>

Universal Objects
Media Objects
- Text, fonts, flow, etc.
- Shapes, Arrows, etc.
- Balloon Help
- Sound and Video Recorders
- Event Theater Recorder
- Movie and MP3 Players
- MIDI
- Sonogram
- World Stethoscope
- Robots

Presentations: books and projects (TNs)

DTP
- **Active Essay**: Weasel

Collaboration
- Sharing Interactions w/ desktop and VOIP
- Email
- Chat, Drag & Drop
- NetMorphs

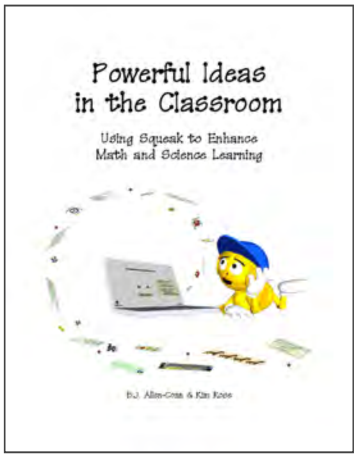
Browser

3D Croquet

Squeak Media
- Pier
- Seaside
- Swiki
- …

Educational Examples

Here are selected examples of the Etoys content we use to help children learn "powerful ideas" in math and science. They are organized roughly by age, but occasionally will show younger children using a computer tool made for them by older children (as in the "Function Machine" for 1st graders made for them by 4th or 5th graders).


Powerful Ideas in the Classroom
Using Squeak to Enhance Math and Science Learning
B.J. Allen-Conn & Kim Rose

Most of the examples (and most of our worldwide experience) has been with children in the 4th through 6th grades, but a number of schools (especially in Japan) have done extensive classroom work with 2nd and 3rd graders, and others (in Canada and in parts of the US) have concentrated on 7th and 8th grades.

The standard starter book for most children and adults is "Powerful Ideas in the Classroom" by teacher B-J Allen-Conn and Kim Rose (Executive Director of Viewpoints Research). We will not attempt to replace that book here, but will draw several of our examples from it in a more summarized form.

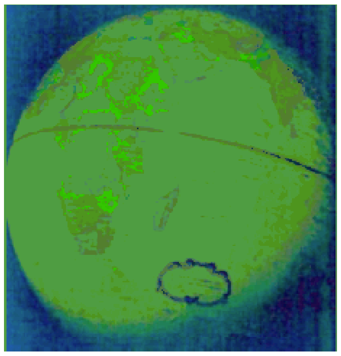
Here we present, not a curriculum, but instead a gist of what a K-8 (or beyond) curriculum in math and science might be like if aided by the authoring by students of their own dynamic simulations of ideas.

The Balinese were once asked why they had no word for "art" – they said "but everything we do is art!" Similarly, pretty much everything children do is art: they make not just to have and for pleasure, but to learn and understand.

Mathematics is the Art of "careful reasoning about how representations of ideas can be related to other representations of ideas". Bertrand Russell famously said, "math is just *p implies q*", but this is like defining Music as "various ways to make noise over time"! We humans, and especially children, above all things are *alive*: we learn best by experiencing being alive while trying to understand. As Goethe put it: "Gray is all Theory, But Green grows the Golden Tree of Life!"

So it is the Romance, Beauty and Emotional Satisfaction that we have to bring home to children. It is along these dimensions that we should help them travel.

Science is one of the Arts of "making the invisible more visible" – not just things our senses can't detect, but also to make visible the causes of the phenomena in the universe we live in. Where Mathematics and other representations of ideas in our minds can be quite arbitrary – they are all essentially stories, where the mathematical stories strive to be consistent – and thus may have nothing to do with our universe even if we love the story, Science tries to find ways to relate "what's out there" to the representations our minds and inventions limit us to.



Mathematics can stand on its own as an art form – like other stories it just needs to follow its own esthetics – but when it is used within Science it acts as a kind of mapping language for territories that can't be seen directly. The pocket globes in the 1780s showed what the Earth must look like from space as a result of the first round of careful geometric(literally *earth-measuring*) surveys and inferences from those surveys – this was math being used to map investigations of "what's out there". It is easy to see that the "map is not what is being mapped", both in kind and in detail. However, since we can only think in terms of our own representations, we often can get fooled (even scientists) when we mistake things in language for the things "out there".

Einstein recommended that "thought experiments" and mathematical formulations should often be done before the observations because the math provides a language to think in and make careful guesses with. We can see this is a sword with two edges: the other edge is that doing lots of math beforehand can prejudge what is investigated if the "math story" is particularly compelling.

In any case, the most important parts of science for children are those that involve direction investigations with the fewest stories. Part of the skills of science are to be able to avoid being ruled by "story thinking" and, given how strongly we are set up by Nature to think in terms of stories, this requires a lot of practice!

Young Children and Mathematics

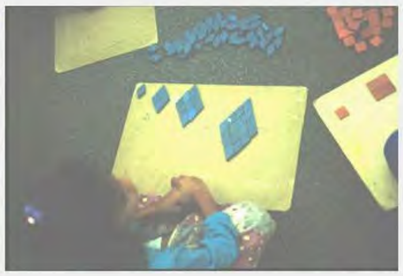
It's a cliché that "children are surprising" but we were hardly prepared for what 6 year-olds can really do when in a rich educational environment. This, and many other encounters like this one, changed our clichéd perceptions of what children can and cannot do. The most important principle in early children's learning is to try to find out what they can do, and what representations of ideas work best for them.

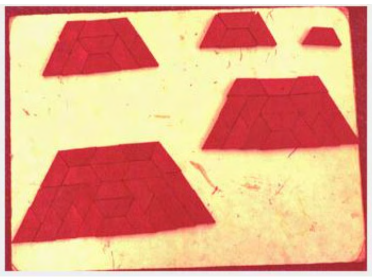
Their teacher, Julia Nishijima, whom we met at one of the schools we worked with 15 years ago, was a little unusual in that she was a natural mathematician.

We don't think that she had studied math formally or had actually had ever taken a calculus course. But she was like a talented jazz musician who has never taken formal lessons. She really understood the music of mathematics.

She had a natural mathematical outlook on the world and here's one of the most interesting projects we saw in her classroom. She had the children pick a shape that they liked, and the idea was to, using just those shapes, make the next larger forms that had the same shape.

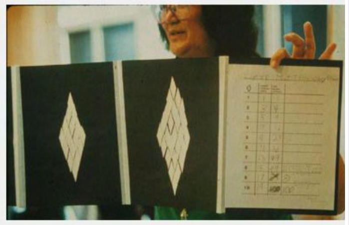
Here are diamonds, squares, triangles. Trapezoids are kind of challenging, you have to turn them around to get them to fit.

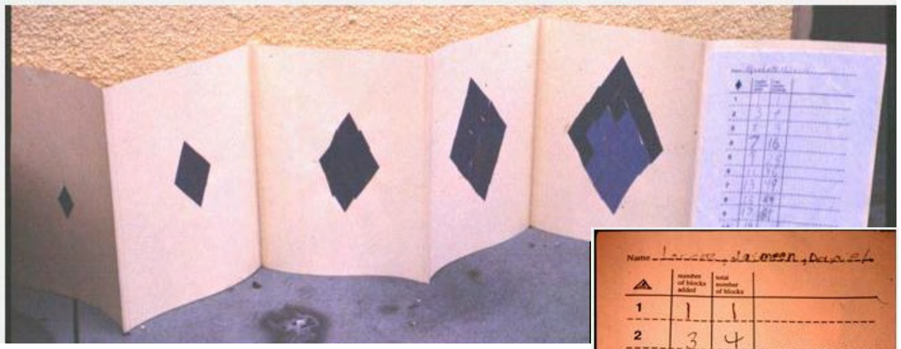


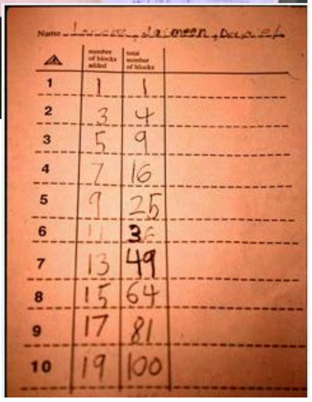


One of the things that this teacher did with her students was to get them to reflect on their creations. She treated math as a kind of a science to get the children to create structures - mathematical structures - and then analyze them. Here's what 6 year old Lauren wrote down that she had noticed.

They cut out cardboard shapes to make a project for their parents. Lauren noticed that it took one tile to make the first one, the total number of tiles was one. It took three more tiles to make the next shape, and the total number of files was four.







Name [illegible]

	number of blocks added	total number of blocks	
1	1	1	
2	3	4	
3	5	9	
4	7	16	
5	9	25	
6	11	36	
7	13	49	
8	15	64	
9	17	81	
10	19	100	

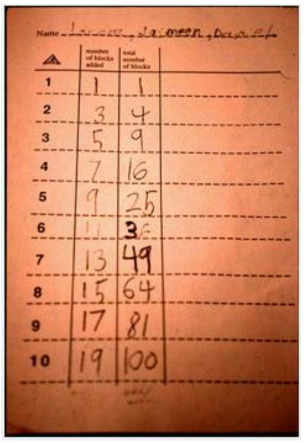
And it took five more tiles to make the next one. So pretty soon she saw "Oh yeah, these are just the odd numbers, differing by two each time, I add two each time and I'll get the next one of these".

And the sum of these is the square numbers, up to at least 6 times 6 here (she wasn't quite sure about seven times seven).

She had discovered two very interesting progressions that all the mathematicians and scientists in the audience will recognize.

Then the teacher had the kids all bring their projects up to the front of the room and put them on the floor so they could all look at them, and the kids were absolutely amazed because all of the progressions were exactly the same!

Every child had filled out a table that looked like this and it meant that the growth laws for all of these were exactly the same and the children had discovered a great generalization about growth.



Name [illegible]

▲	number of blocks added	total number of blocks	
1	1	1	
2	3	4	
3	5	9	
4	7	16	
5	9	25	
6	11	36	
7	13	49	
8	15	64	
9	17	81	
10	19	100	

The mathematicians and scientists who are reading this will recognize that the odd numbers are produced as a first order differential relationship that gives a smooth, uniform progression – in the computer mathematics that we use, this is expressed as *doing over and over* with *increase-by*:

Doing over-and-over: *Odds* increase-by 2

And the Total number of tiles is produced by a second order (because it uses the results of a first order relationship) differential relationship:

Doing over-and-over: *Totals* increase-by *Odds*

The idea of *increase-by* is easy for everyone, because it is just putting things (or adding things) into a pile of things. Now what's wonderful about this is that these sequences the first graders discovered can be used to represent a lot of classical science, including uniform growth and constant acceleration.

And since these are made only out of addition (because they're computed incrementally), it occurred to Seymour Papert and others that young children should be able to deal with this kind of math – especially as it applies to the physical world – much more powerfully than the standard kind of calculation that they are subjected to in schools: and this turns out to be the case. Later we'll see that the circle drawn by a Logo turtle is a first order doing over-and-over and increase-by where the "numbers" are like arrows that have both length (size) and a direction.

Over the years, a number of researchers have looked deeper into "what children can do", and found that the reality is quite different than most adults, and especially formal schools, suppose. The important "catch" as pointed out by Montessori, Piaget, and others is that children simply do not think the same way throughout their lives: they have particular styles of thinking in different parts of development and can learn very powerful ideas if they are put in a suitable form – or they can suffer terribly if forced to learn ideas in a form that their minds are not yet ready for. Most especially, the school approaches of: (a) knowledge is a fluid that can be poured in the child's ear by a teacher, and (b) that children are defective adults who need to be fixed, are both doomed to failure both during application, but also often failure in later life as well.

Following are a number of examples of what children can do when put in a rich environment that caters to the ways their minds work. And just as Papert suggested in the 1960s, the computer can be shaped into one of the most compelling and powerful educational environments in human history. This shaping is still ongoing, but enough experience has been gained to make strong arguments for very different forms of education in the future.

Mathematics For Children

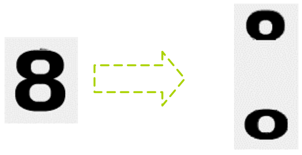
Mathematics is basically "careful thinking about how one collection of representations could imply another collection of representations" – or, as Bertrand Russell famously put it, "math is just *p implies q*". But this is like describing Music as "various ways to make noise over time"! As Goethe remarked: "Gray is all Theory, But Green grows the Golden Tree of Life!"

Both Music and Mathematics are Art Forms that have much Romance, Beauty and Emotional Satisfaction as their main reasons for being. It is along these dimensions that we should help learners travel.

So, for almost everybody, especially children, what we really want to do is to put learners in an environment in which mathematics is the way interesting and beautiful things get done and thought about, and where the special character of mathematics is gradually revealed.

Another idea that underlies mathematics is the notion of taking things apart (analysis) and putting them together (synthesis). From this we easily get the ideas of *quantity* and *counting* and we start to get interested in the idea of *numbers* and how they can be taken apart and put together. If we take the number "8" as an example, then we quickly see that "the idea of eight" is very different from the many ways we can *represent* "the idea of eight". So we have numeral forms (names for numbers) such as: 8, VIII, etc. And we have various analogous forms, such as ||||||||, talleys, groupings by 5s, lengths, etc.

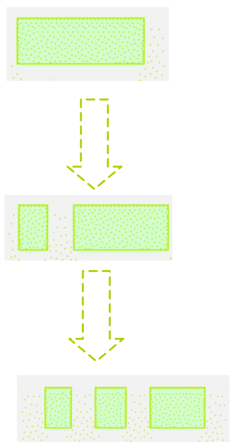
Since we are interested in helping children learn to think about how to take numbers apart and put them together, we would like to choose representations for numbers that encourage powerful thinking. We can see that the official Arabic form of "8" is not a very good representation for this. It is compact, but it is for already expert numerate thinkers. Why is this? Because "8" has no simple "take apart" properties. We can split it into two "o" s or a forwards 3 and a backwards 3, but neither helps us think about "8".



On the other hand, the simple talley |||||||| does have nice "take apart" properties. We can easily separate it into all the whole number sums and products that make it up.

| ||||||| || |||||| ||| ||||| |||| |||| ||||| ||| |||||| || ||||||| | and so forth.

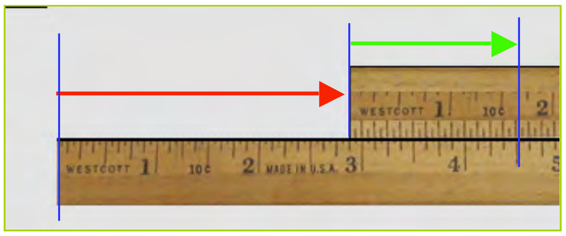
|||| || ||
|||| || || etc.



So, we strongly recommend that children be started with such analogical representations for numbers, and that these be used for thinking with, even after we start to label them with standard "numeral names". Quite a bit of the difficulties most people have with mathematics (and especially with numbers) arises because standard schooling is in a rush to get children into the adult conventions (even if this rush causes them to loose mathematics for the rest of their life).

An even better representation for a number is a *length* of material that can be compared to other lengths, cut in pieces and put back together anyway we want. So we could think of strands of plasticine that can be cut anywhere and joined any way.

A more durable version of the plasticine is a ruler, which indeed uses lengths as analgous to numbers. If we have two rulers we can make a universal adding machine for all kinds of numbers, including fractions. Below: 2 7/8 + 1 5/8 = 4 ½ .



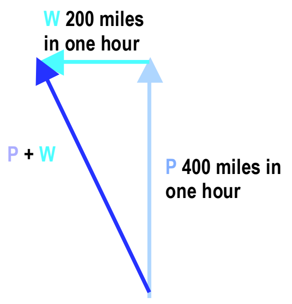
This way of thinking about numbers is not only simple and useful, it is very much like the way scientists think of numbers.

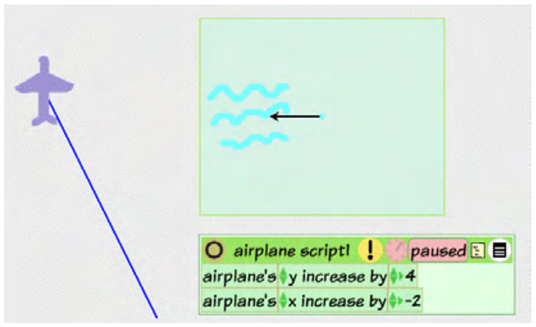
For example, suppose we think of lengths as getting larger as *increasing in a direction that moves towards the right*. We think of "addition" as placing one length to the right of another. The combined length is the *sum* of the addition. And we can find out what it is by measuring it with another length. So it is natural to think of subtraction as joining one length with another that is aimed in the opposite direction. The resulting length can be similarly be measured by another length.

We can combine these two ideas by putting an arrow on our lengths to indicate which way they are pointing. Then we can see that "subtraction" is just "addition", if "addition" means *join the tail of our second number to the arrow of the first number and then measure to the arrow of the second number to get the result*. This seems almost ridiculously simple, but it is very powerful! It is visual and geometric, and it allows us to do a lot of complex sums (including fractions) without having to learn the "base 10 positional notation algorithms" that use up so much of most children's time and spirit.

Another powerful use of this representation is that it works not just in one dimension (of "regular numbers") but it also works in two dimensions, 3D, and even any number of Ds. We will take this use up later on in this book, but for now let's think about how we would calculate a plane **P** traveling at 400 miles an hour heading N but flying in a 200 mile an hour wind **W** that is heading W. The distance is just **P**+**W**.

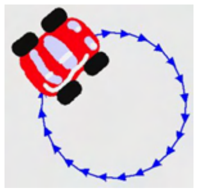
We can make a simple Etoy to see what happens by moving N at rate 4 and having the wind blow W at rate -2 over and over.




```
airplane script1   paused
airplane's y increase by 4
airplane's x increase by -2
```

The plane is heading N (so is pointing up) but is being blown to the W. The moment by moment path is thus a diagonal.

This is a very useful way to think about numbers – as directed lengths (arrows) and addition. And this is one of the main ways we would like children to think about numbers: because it is simple, powerful, and *because this is one of the main ways that science thinks of numbers* (numbers that are used in this way are called "vectors").



Another powerful idea is to describe change as incremental in time and/or extent. For example, the more complicated description of the total number of tiles as squares of the step number in the growth of shapes project by the 6 year olds could be replaced with the much simpler idea that the change of the amount of new tiles that had to be added was always just 2. A turtle circle is the sum of a series of incremental vectors.

This is called a "differential model" and the written down forms of these models are called "differential relationships", and are the heart and soul of the calculus. Historically, most of them have been evaluated using algebra, but they can also be done as simple incremental sums, and this is just what computers are really good at. The great benefit here is that this way of thinking, which is so central to much of science, can now be learned by young children long before then can get fluent in algebra. We will include many examples of how the children are able to find and use these.

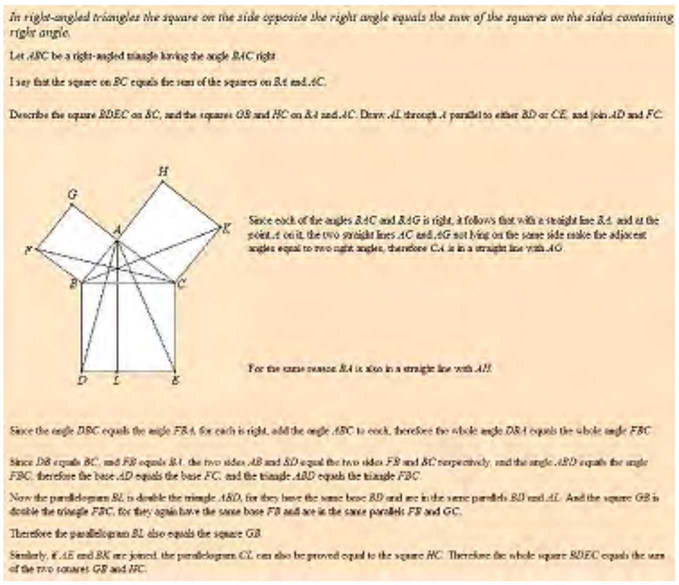
Mathematical Reasoning

We characterized mathematics as "careful thinking about how representations of ideas could imply other representations of ideas", and that the most important process in helping anyone learn how to do mathematical thinking is to put them in many situations in which they can use how they think right now in a more careful way.

For example, what is a really good representation to help children think about the Pythagorean Theorem?

Below left is Euclid's proof of the Pythagorean Theorem for high school geometry students. It's elegant and subtle, it casts light on other areas of geometry, but it is not a suitable first proof for most young minds.

Below right is a very different kind of proof: perhaps the original one by Pythagoras. We have seen many elementary-age children actually find this proof by playing with triangle and square shapes. Show the arrangement, surround the C square with 3 more triangles to make a larger square, copy the larger square, remove the C square, rotate the two triangles, note there is room for the A and B squares, move them, and Bingo! This proof has a visceral approach and feel, a *powerful simplicity* that is perfect for beginners' minds and provides a solid foundation for later more abstract and subtle looks at the idea.



In right-angled triangles the square on the side opposite the right angle equals the sum of the squares on the sides containing right angle.

Let *ABC* be a right-angled triangle having the angle *BAC* right.

I say that the square on *BC* equals the sum of the squares on *BA* and *AC*.

Describe the square *BDEC* on *BC*, and the squares *GB* and *HC* on *BA* and *AC*. Draw *AL* through *A* parallel to either *BD* or *CE*, and join *AD* and *FC*.

Since each of the angles *BAC* and *BAG* is right, it follows that with a straight line *BA*, and at the point *A* on it, the two straight lines *AC* and *AG* not lying on the same side make the adjacent angles equal to two right angles, therefore *CA* is in a straight line with *AG*.

For the same reason *BA* is also in a straight line with *AH*.

Since the angle *DBC* equals the angle *FBA*, for each is right, add the angle *ABC* to each, therefore the whole angle *DBA* equals the whole angle *FBC*.

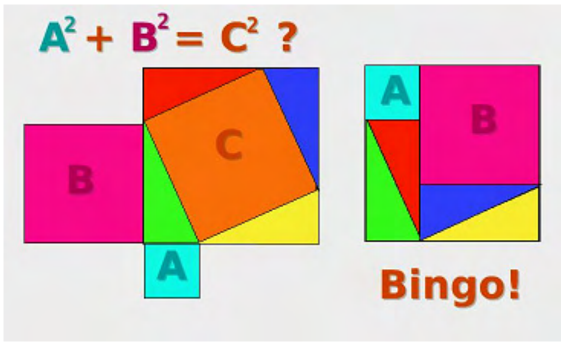
Since *DB* equals *BC*, and *FB* equals *BA*, the two sides *AB* and *BD* equal the two sides *FB* and *BC* respectively, and the angle *ABD* equals the angle *FBC*, therefore the base *AD* equals the base *FC*, and the triangle *ABD* equals the triangle *FBC*.

Now the parallelogram *BL* is double the triangle *ABD*, for they have the same base *BD* and are in the same parallels *BD* and *AL*. And the square *GB* is double the triangle *FBC*, for they again have the same base *FB* and are in the same parallels *FB* and *GC*.

Therefore the parallelogram *BL* also equals the square *GB*.

Similarly, if *AE* and *BK* are joined, the parallelogram *CL* can also be proved equal to the square *HC*. Therefore the whole square *BDEC* equals the sum of the two squares *GB* and *HC*.

This?



Or This?

The rule of thumb here is to find many ideas and representations for them that allow "beginners to act as intermediates", that is, for them to immediately start doing the actual activity in some real form. This builds both skill and confidence, and both of these are what is needed to happily forge through more subtle and complex ideas.

The way calculus looks at many ideas is so powerful and so important that we want to start children learning to thing along these lines much earlier in their lives. So we have made a form of real calculus that is thinkable by young minds and that the computer brings alive in many delightful ways.

We will show this approach mostly in examples that follow, but it is based on the kinds of arithmetical and mathematical reasoning that we have seen very young children do in 1st grade (and even younger in the wonderful kindergartens of Reggio Emilia). The basic idea is that many relationships of change can be characterized by simple accumulative additions, sometimes cascaded. These generate all the smooth ramps and parabolic relationships that can represent many of the findings of science. This insight is also what got Babbage started thinking about making machines that could compute.

A Few Examples For Young Children

Young children get a lot of their best learning from direct contact and manipulation of things in the physical world. For many reasons, television is one of the worst mediums we can put in a child's world. Thinking of it as an unlocked cabinet full of loaded guns and deadly poisons is a little simplistic, but does suggest that it should not be available to children, or at least should be locked up.

On the other hand, we want to have books in the children's world, and we would like to do as much as possible to motivate and help children to read.

The computer is a little trickier, in part because it can easily imitate television and TV-like media, but it can also imitate books (and, even better, dynamic books).

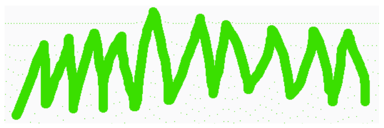
So a healthy balance is called for between direct contact with objects and the indirect contact with symbolic objects that books and computers provide.

One of the general rules of thumb for 21st century children is that we want to help them escape from the very simple perception of the world that their senses (including their "commonsense") provides them. When adults go on walks with children, they should always carry an inexpensive magnifying glass and telescope. The children will easily get used to the idea that there are always things in their environment that their unaided senses are missing. In other words, "the world is not as it seems", and this is the modern version of "open sesame!" that leads to science.

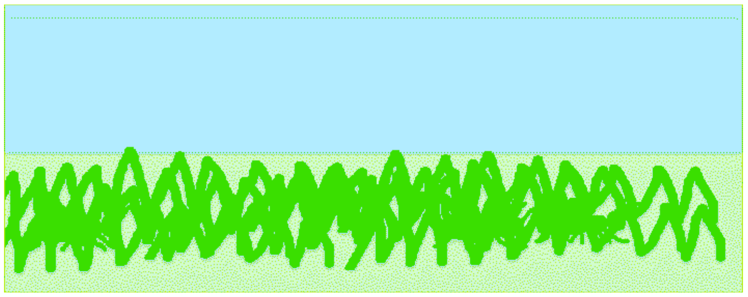
An Example From Nature: Camouflage

Camouflage is found everywhere in nature, as is the tendency for many animals to stop still when alarmed. A fun project for a young child is to make a camouflage project and see a plausible explanation of why animals "freeze" when threatened.

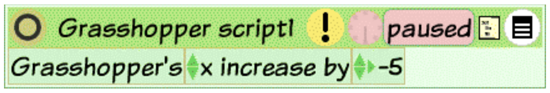
First, let's paint some grass, and then paint a grasshopper using the same shade of green.



If we put the grasshopper in the grass, we see right away why its green color might help it escape from birds and other predators that might want to eat it. (If a creature like a grasshopper lived in a desert environment, what color might we expect it to be?)



Now let's write the simplest of scripts for the grasshopper to get it to move sideways in the left-right direction (the official name for this direction is **x**). We see that this is the same as the rule the children found for steadily increasing the size of something:



```
Grasshopper script1   paused
Grasshopper's x increase by -5
```

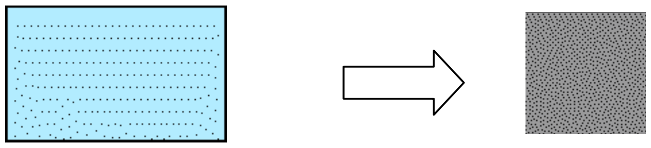
All of a sudden we can see where the grasshopper is, and so can most other animals (too bad this document isn't dynamic!). Now we have a little insight into why animal vision is almost always better at detecting motion than in seeing fine details – and also, why animals might want to keep still when threatened.

Illusions

Illusions furnish many examples of "the world is not as it seems", and many of them are quite startling when done dynamically using the computer. And, some of them can most easily be made using the computer.

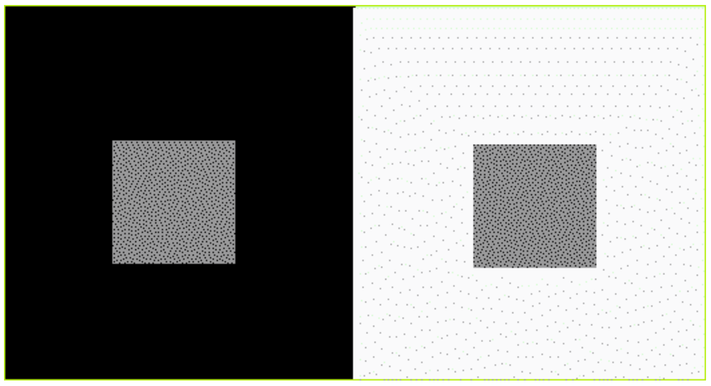
Contrast Illusions

When we look at a scene, we think we are seeing the lights and shades and colors accurately, but in fact, both our lower level and higher level mental mechanisms are reformulating what our retina senses. For example, the apparent brightness of a gray tone is quite conditioned by the brightness of the background around it. We can show this both statically and dynamically. Let's get out a rectangle from the Supplies Bin and color it.



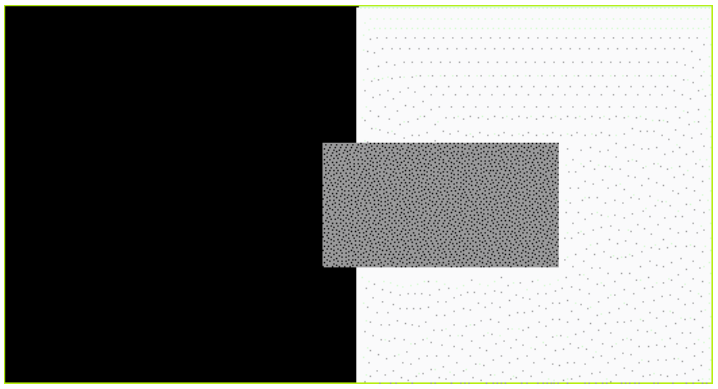
We make a square from it and color it a neutral gray, and then copy it so we have two of them. Then we make a larger square, color it black, and then copy it and color the copy white.

Then we drop all of these into a playfield rectangle (because it is a little self-contained world and is set up to "capture" objects dropped on it). We organize the rectangle so it looks like:



We can already see that the right hand rectangle on the white background looks darker than the copied rectangle with the same gray color looks on the black background. This is called the "Mach Illusion", and it serves to help animals to see things that are almost the same color (because it makes the difference seem more different).

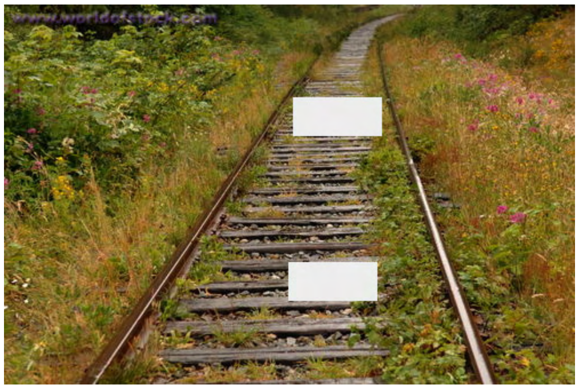
But we can also do a dynamic version that seems to change colors, and also "proves" that the two squares are the same color. We write a script that is a lot like the grasshopper script but we include a "bounce" tile that will bounce off the walls and make a sound (the teacher's favorite sound is the one called "silence").



Illusions That Teach

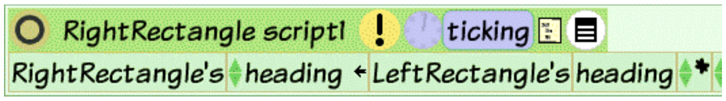
The “railroad tracks” illusion is one of serveral that supposedly only work with human beings that have been raised in a constructed environment (this is still somewhat controversial) but perhaps the widespread use of the OLPC XO will help answer some of these questions.

The basic idea is that the same sized object placed on a picture of railroad tracks will look larger when placed higher where the tracks appear narrower. The mental calculation is that things of a certain visual angle in a context that whispers “farther away” are perceived to be larger. Here are two rectangles placed on a picture of railroad tracks.



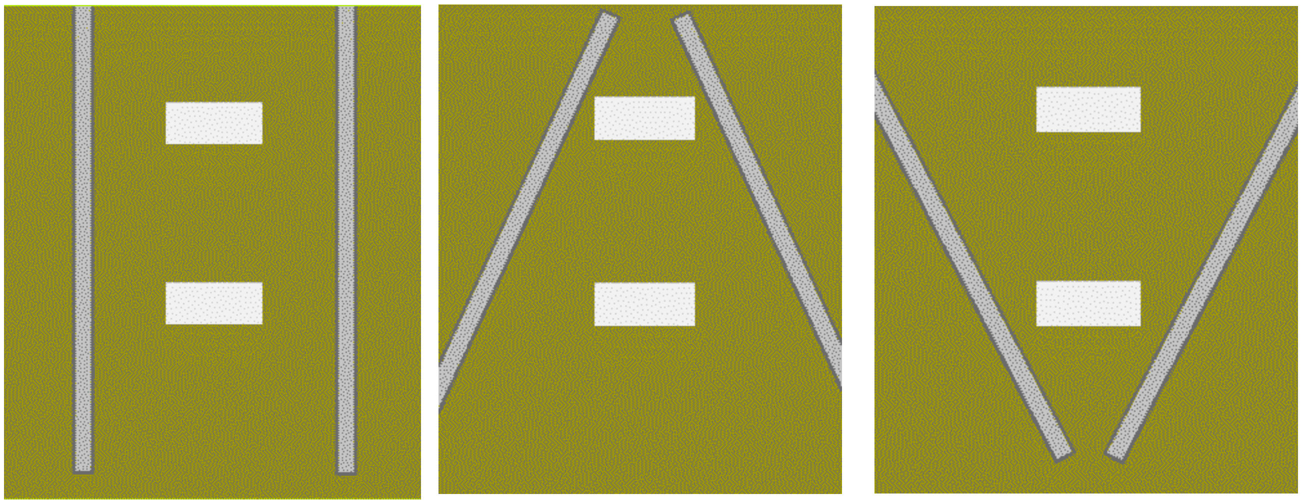
We can make a dynamic version of this illusion in Etoys, and also learn something about angles, negative numbers, and how much convergence is required to manifest the illusion. We get out a rectangle, make it high and skinny, and then copy it.

We want to write a script that will make one of the rectangles rotate in the opposite direction when the other one is turned.



```
RightRectangle script1 ! ticking
RightRectangle's heading ← LeftRectangle's heading * -1
```

So, over and over again, whatever we rotate the left rectangle to, the right rectangle will be rotated to the opposite angle.



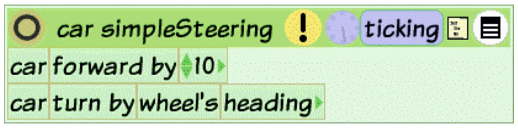
We see that the “enlarge effect” is similar to that in the picture of the railroad tracks, and that there is very little effect when the Etoy “tracks” are rotated outwards.

To Drive And Steer The Car (Learning About Naming & Variables)
To drive the car, the children find that changing the number after **car turn by** will change its direction.

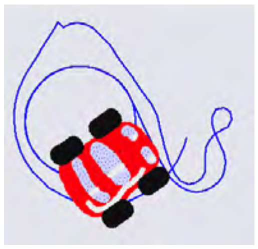
car simpleSteeri

Then they draw a steering wheel (the very same kind of object as the car, but with a different costume) and see that if they could put **steer's heading** right after **car turn by …** this might allow the steering wheel to influence the car.

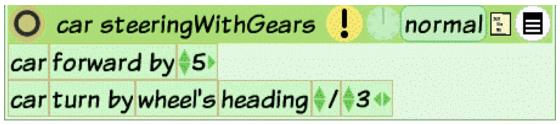
They can pick up **steer's heading** (the name for the heading numbers that the steering wheel is putting out) and drop it into the script. Now they can steer the car with the wheel!


```
car simpleSteering ! ticking
car forward by 10
car turn by wheel's heading
```

The children have just learned what a variable is and how it works. Our experience indicates that they learn it deeply from just this one example.



They quickly find that it is hard to control the car. They need to introduce a "gear" into the wheel's connection to the car. They can get the needed advice from a teacher, parent, friend, or from a child thousands of miles away via the mentoring interface over the Internet. They open the expression in the script, and divide the numbers coming out of the steering wheel by 3. This *scaling* makes turns of the steering wheel have less influence. They have just learned what divide (and multiply) are really good for.


```
car steeringWithGears ! normal
car forward by 5
car turn by wheel's heading / 3
```

The Powerful Idea of Feedback

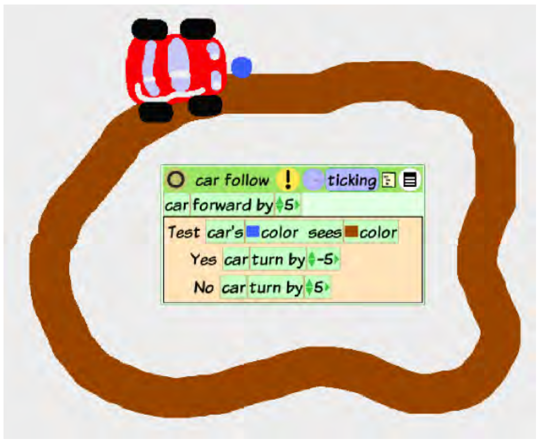
Sometimes we have enough information to make a foolproof plan. But most of the time things don't go quite as expected, (even with our "foolproof" plans), and we wind up having to find new information, make new corrections, sometimes new plans.

All animals and other mechanisms have limited abilities to gather information and very little ability to extrapolate into the future. For example, the simplest bacteria can be hurt or killed by too much acidity (or too little). They have evolved molecular machines that can help them detect when some dangerous substances are starting to affect them, and the ones that swim exhibit a tumbling behavior that radically (and randomly) changes the direction in which they are swimming. If things are "good" they don't tumble, if "bad" then they tumble again.

This general strategy of sensing "good" and "bad" and doing something that might make things better is pervasive in biology, and is now in many of the mechanical and electrical machines made by humans.

A fun activity for children is to be challenged to find their way around the outside of one of their school buildings blindfolded and just by touch. The simplest strategy works, follow the wall, and always turn in the same direction when it is lost.

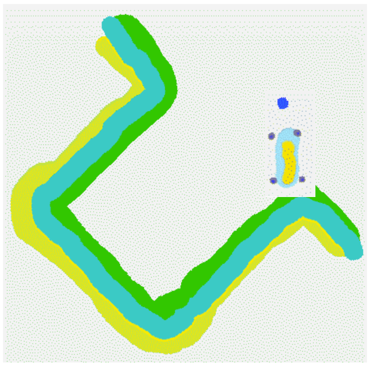
We can get our cars to do that by painting a different color to be used as a "touch sensor" and writing a script that looks like:

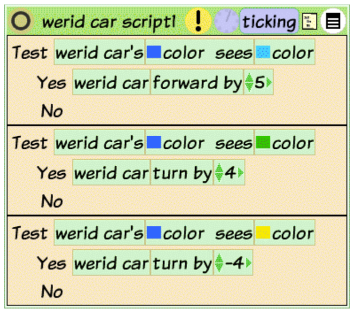


```
car follow ! ticking
car forward by 5
Test car's 🟦color sees 🟫color
   Yes car turn by -5
   No car turn by 5
```

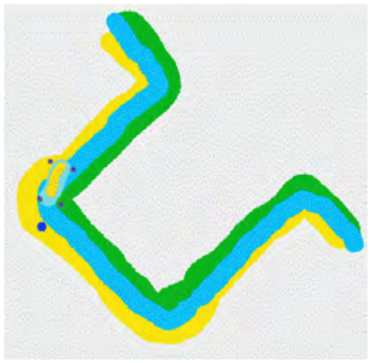
Then we challenge the children to make a car and a road that will get the car to go down the center of the road instead of the outside. There are many solutions for this. Here is a nice one from two 11 year old girls who worked well together.

They figured out that if they made curbs on the road of two different colors then there would be only three cases: when the sensor is on the center, or one of the two sides. Their car, road and script look like:





```
werid car script1 ! ticking
Test werid car's 🟦color sees 🟦color
   Yes werid car forward by 5
   No
Test werid car's 🟦color sees 🟩color
   Yes werid car turn by 4
   No
Test werid car's 🟦color sees 🟨color
   Yes werid car turn by -4
   No
```

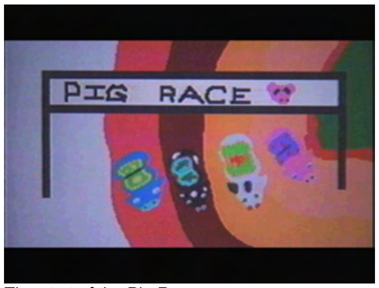


We can see that this is a better script than the one we showed them. They decided that their robot car would only go forward when it is in the middle. This means that it can safely negotiate any turn (the example can't).

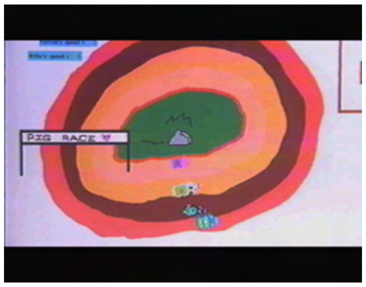
Jenni's Big Race, … err, Pig Race

Jenni, Age 11

Then we turned the kids loose to think up a project all by themselves. Jenny liked to introduce her slant on things whenever possible (her cars we done as pencils with wheels), and she decided that it would be really fun to make a pig race.



The start of the Pig Race

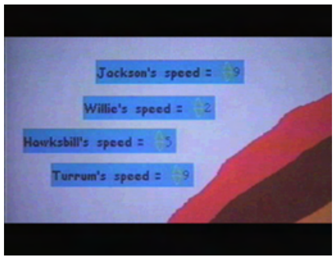


The race. How does Jenny get the pigs to stay in their lanes?

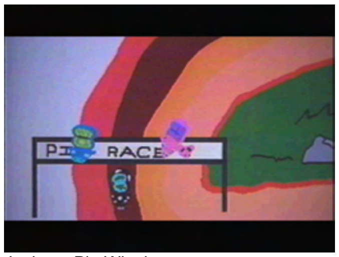
Here is her narrative as she explained her project for the Squeakers DVD.

> Here we go for the annual pig race. It looks like it's a rough time today. The pink pig crashing into the wall.
>
> (I also have a watcher here that tells me what speed they're going.)
>
> Oh and the blue ;pig's coming in the lead. I have my own pig. I named it Jackson. And usually it loses, But, oh look the black pig is catching up, leaving the white pig in the dirt!

She wanted a real track for her pigs and needed to figure out how to keep each pig in its lane. She had a wonderful idea that the nostrils of her pigs would be perfect as sensors to tell when it was trying to escape its lane!


Jackson's speed = 9
Willie's speed = 2
Hawksbill's speed = 5
Turrum's speed = 9

Watchers showing the speed of the different pigs

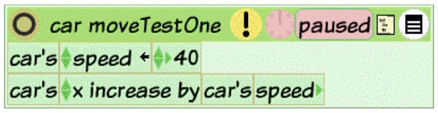


Jackson Pig Winning

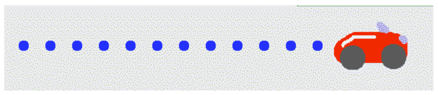
Reflections on Speed

Quite a bit of doing is "just doing", so it is a good idea to also reflect on what just happened. One way to do this is to have the objects leave trails that show what they were doing over time.

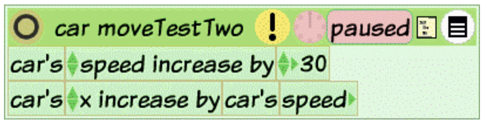
If the speed is constant then the trail of dots is evenly spaced, showing that the same distance was traveled in each little tick of time



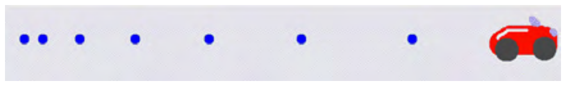
```
car moveTestOne  paused
car's speed ← 40
car's x increase by car's speed
```



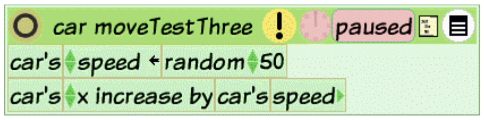
If we increase the speed each tick of the clock, we'll get a pattern that looks like the second picture. This is the visual pattern for uniform acceleration



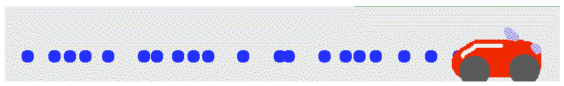
```
car moveTestTwo  paused
car's speed increase by 30
car's x increase by car's speed
```



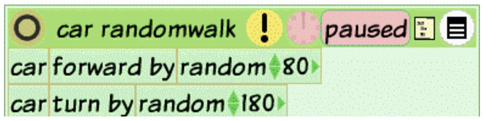
If we make the speed be random each time (in this case between 0 and 40), we will get an irregular pattern of distance traveled for each tick



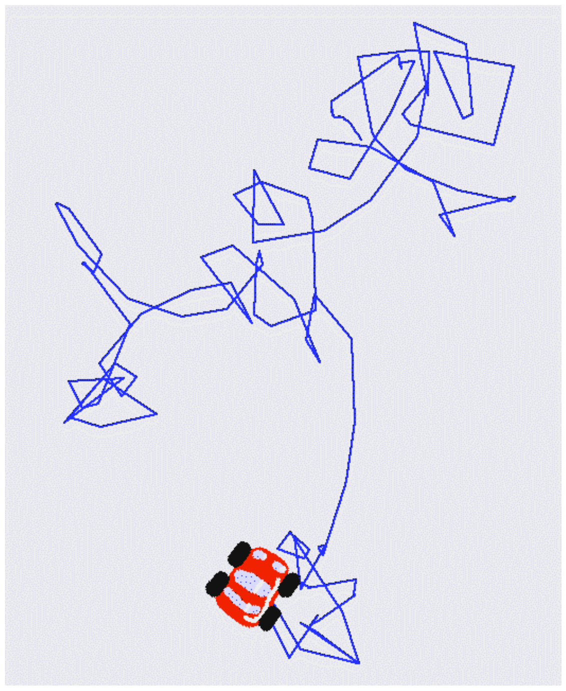
```
car moveTestThree  paused
car's speed ← random 50
car's x increase by car's speed
```



It's fun to try using random in two dimensions. Here we can move a car forward randomly and turn it randomly. If we put the pen down, we get a series of "Drunkard's Walks".



```
car randomwalk  paused
car forward by random 80
car turn by random 180
```



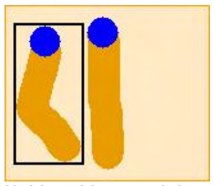
One way to look at this is that a little bit of randomness will cause a lot of territory to be visited given enough time and it changes the probability of collisions quite a bit (it is still low, but now "more possible").

So we might guess that a random walk in a script that is using the principles of feedback could accomplish surprising things.

Animations

Now let's look at the simple animations done by children. In most productivity media systems animation is a built-in and opaque feature, but in Etoys it is something that is actually constructed by the children because there are many powerful ideas associated with animation-like processes.

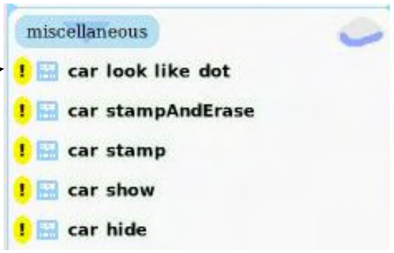
Car made by a child


Holder with several drawings of a worm

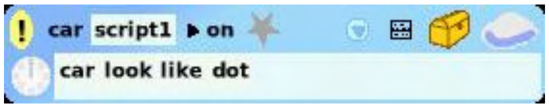
To do animations we need to use a variety of costumes that can be changed via a simple script. For example, let's make a few drawings of a worm and put them in a holder. If we look in the car's viewer we find

car look like dot

and drag it out to start a script.


miscellaneous
car look like dot
car stampAndErase
car stamp
car show
car hide

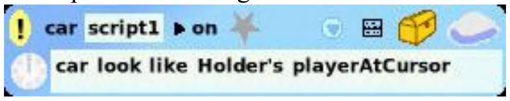
This viewer category contains properties and behaviors that deal with an object's appearance.

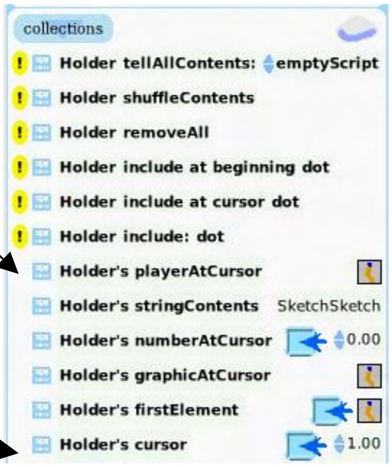

car script1 ▸ on
car look like dot

We want it to look like one of the drawings in the holder so we look in the holder's viewer and find

Holder's player at cursor

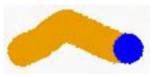
We drop it on "dot" to get:


car script1 ▸ on
car look like Holder's playerAtCursor

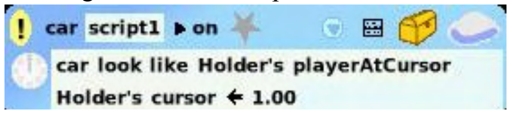

collections
Holder tellAllContents: emptyScript
Holder shuffleContents
Holder removeAll
Holder include at beginning dot
Holder include at cursor dot
Holder include: dot
Holder's playerAtCursor
Holder's stringContents SketchSketch
Holder's numberAtCursor 0.00
Holder's graphicAtCursor
Holder's firstElement
Holder's cursor 1.00

This viewer category contains properties and behaviors what an object is carrying.

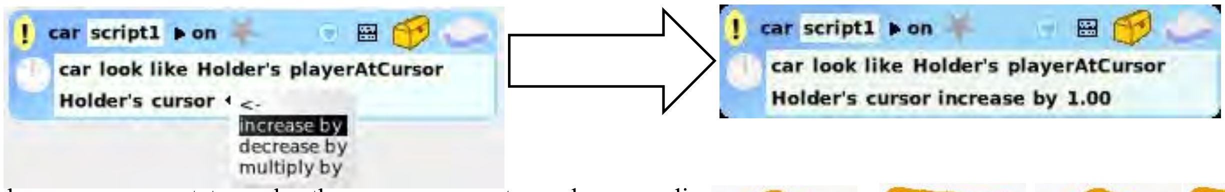
We click on the (!) to try this script and we see that the car's costume is changed to a worm.



But further clicks won't cause a further change because we haven't moved the cursor. To do this we look in the Holder's viewer again, find the cursor change line and drag this into the script.

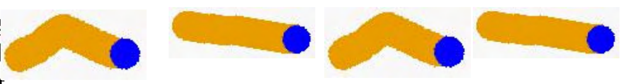

car script1 ▸ on
car look like Holder's playerAtCursor
Holder's cursor ← 1.00

Finally, we change the <- to "increase by" by clicking on it.

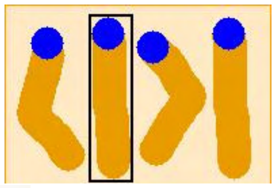

car script1 ▸ on
car look like Holder's playerAtCursor
Holder's cursor ← <-
increase by
decrease by
multiply by

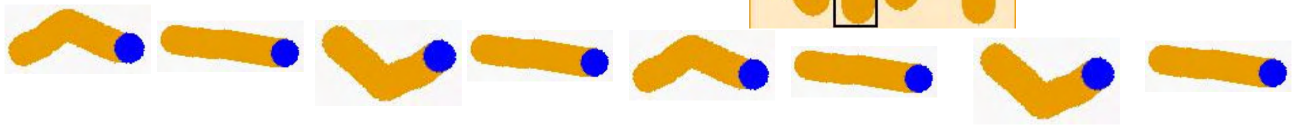
car script1 ▸ on
car look like Holder's playerAtCursor
Holder's cursor increase by 1.00

because we want to make the cursor move to each succeeding position in the Holder on each tick of the clock (the Holder will wrap around these numbers when we get to the end). We set it ticking and we get a nice worm animation.



We can make more worm pictures and drop them into the holder (even while it is running) to get a smoother animation look..

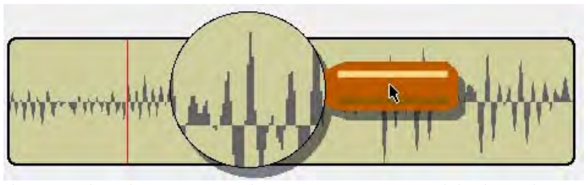




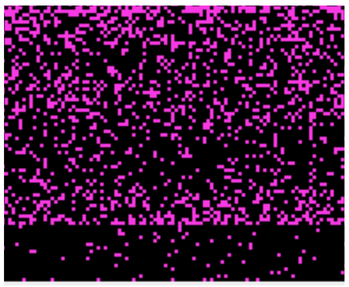
Sound Synthesis is “Just Animation” with Increase-by

The Etoys Sound Recorder will capture a sound and automatically put it in the Sound Library where it can be played as a sound effect by an object.

But we can also look at this sound. It is held by Etoys as many tiny rectangles in a holder that works the same way as the animation holder: there is a cursor, it can be positioned, etc.



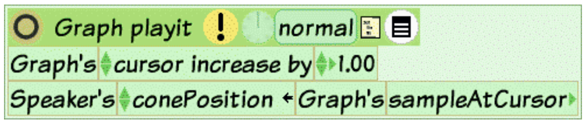
These holders are long. The tiny rectangles are not very wide but there may be as many as 8000 of them for each second of recorded sound! The height of the tiny rectangle is how far the sensor in the microphone was moved when being hit by air molecules (think of this sensor as being like a piston being hit with a shockwave of particles). We will look more at soundwaves later. For now, we can think of them as being made from little shockwaves of bunched and sparse particles moving outwards from the sound source.



If we could waggle the loudspeaker in our computer according to the heights of the rectangles in order, we should be able to hear the sound played back!

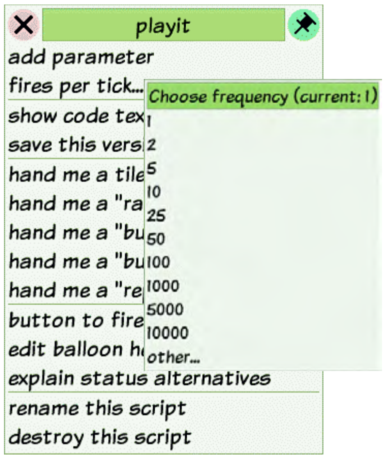
Etoys has a little speaker object that is hooked to the physical speaker in the computer. If we move its cone, the real speaker cone will move in the same way.

This gives us a way to think about making a sound synthesizer – we just have to animate the speaker according to the rectangles in order in the holder. This is very similar to the animations we’ve just done. Our script looks like:



```
Graph playit  normal
Graph's cursor increase by 1.00
Speaker's conePosition ← Graph's sampleAtCursor
```

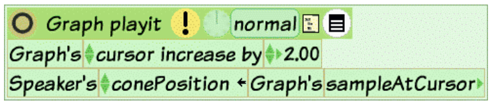
But when we run the script we can’t hear anything. Why is this? It is because the Etoy scripts are set to run at normal animation speeds of several ticks to 30 ticks per second. But the sound fluctuations go much faster. There is a control in each script to set how many times the script should be run for each regular tick. We will go from “1 script firing per tick” to “10,000 script firings per tick” (really fast).



```
playit
add parameter
fires per tick...
show code tex
save this vers
hand me a tile
hand me a "ra
hand me a "bu
hand me a "bu
hand me a "re
button to fire
edit balloon h
explain status alternatives
rename this script
destroy this script
```

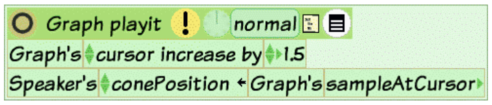
```
Choose frequency (current: 1)
1
2
5
10
25
50
100
1000
5000
10000
other...
```

This works! Now, let’s try some of the same tricks that we used for animation. Let’s set the speed number to 2 (this will pick every other rectangle, and will thus get through the holder of rectangles twice as fast). What does this sound like?

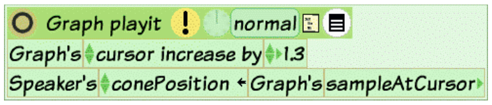


```
Graph playit  normal
Graph's cursor increase by 2.00
Speaker's conePosition ← Graph's sampleAtCursor
```

Now let’s try some fractional numbers. How about 1.5? How about 1.3? Now we can see that there is a “speed number” for each pitch at which we want to play our sample. A commercial synthesizer works exactly the same way.



```
Graph playit  normal
Graph's cursor increase by 1.5
Speaker's conePosition ← Graph's sampleAtCursor
```



```
Graph playit  normal
Graph's cursor increase by 1.3
Speaker's conePosition ← Graph's sampleAtCursor
```

We can make a little keyboard from buttons that will feed in their pitch number and play the sample.

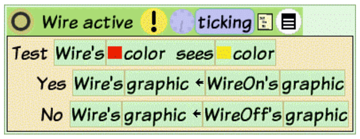
We see again why **increase by** is a powerful idea!

Now, what do you suppose would happen if we used the two stages of **increase by** that produce acceleration? (Answer: we would get what is called “frequency modulation” sound synthesis – another technique that is regularly used by commercial synthesizers.)

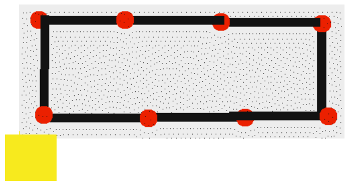
Making A Circuits Simulator Using Costumes

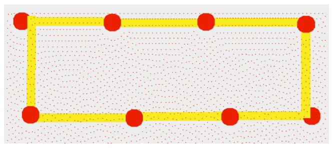
Up to this point we have been using costumes for various kinds of movement and animations. But we can also use them to indicate the *state* (on or off, etc.) of the parts of a simulation. For example, if we wanted to do a circuit simulator that has sources of current, simulated wires, bulbs, and switching components, it would be nice to show what they are doing by changing their appearance.

For example, let's use a yellow color to indicate "current" or, more accurately in this project, "signal". We can make a yellow square as a "source" of this signal. And we can draw a segment of a wire that can propagate the signal. Since the signal can be there or not, we can draw two costumes, one with the wire holding the signal and glowing yellow, and one with the wire without the signal, showing black. We'll put a red connector at one end of the wire that can sense whether a signal is being passed to it. Now we want to give dynamic behavior to the wire. This is a very simple script that uses the same "color sees" test that we used for the path following project. If our red dot can see the yellow signal, then we will switch our costume to the yellow "have signal" or "on" costume. It we can't see the yellow signal, then we will switch our costume to the black "no signal" or "off" costume.



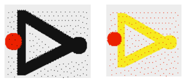
```
Wire active ! ticking
Test Wire's 🟥color sees 🟨color
  Yes Wire's graphic ← WireOn's graphic
  No  Wire's graphic ← WireOff's graphic
```

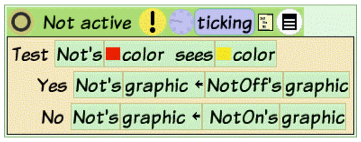




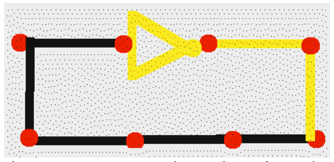
We can copy a bunch of these wires (or better yet, we can put one in the Supply Bin, which will make an unlimited number of copies). If we line them up and touch one end with our yellow rectangle we can see the signal moving down the wires! If we make a loop of the wires, the signal will stay on indefinitely. (In the real world this would only happen if the wires were a superconductor or had some hidden properties, such as a signal strength restorer.)

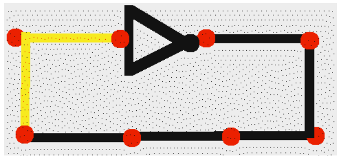
Now we can think about making some "switchable switches" that can sense the signal and do various things with it. For example, one of the simplest "switchable switches" will sense a signal and turn it into "no signal", and sense "no signal" and turn it into "signal". It is called an "inverter". Again, we draw two costumes, one for "on" and one for "off", but the logic is exactly the opposite of the wire. We can use this to make a ticking clock signal by feeding its results back into itself. Notice that the delay made by the propagation of the signal avoids any paradox and simply switches in on and off in order. We could call this a NOT, because it negates its input.



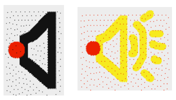


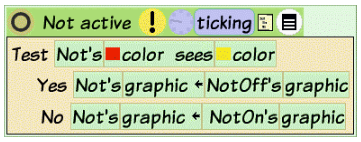
```
Not active ! ticking
Test Not's 🟥color sees 🟨color
  Yes Not's graphic ← NotOff's graphic
  No  Not's graphic ← NotOn's graphic
```



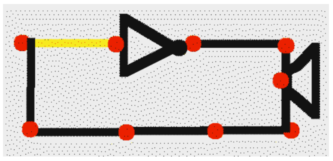


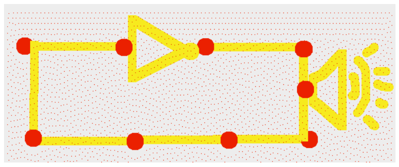
At this point it will be fun to make some output gadgets that can respond to the signal. A light is just a wire that is shaped like a light bulb. Let's also make a loudspeaker that will play a sound if it detects a signal. This will look and act like:



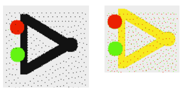


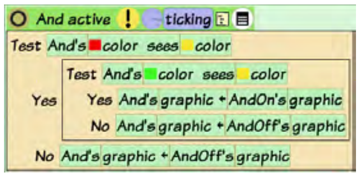
```
Not active ! ticking
Test Not's 🟥color sees 🟨color
  Yes Not's graphic ← NotOff's graphic
  No  Not's graphic ← NotOn's graphic
```



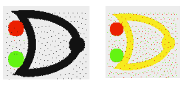


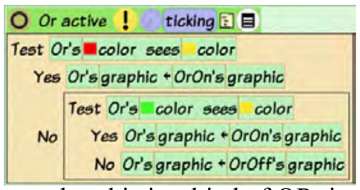
Now we can make some "switchable switches" that can compare signals. For example, we would like one that will only respond with "on" if both of its inputs are "on". Otherwise, it will respond with "off". This is called an AND switch. Switchable switches are sometimes called "gates" (because they "gate" the signal one way or another) so we can also call this an AND gate.





```
And active ! ticking
Test And's 🟥color sees 🟨color
  Yes  Test And's 🟩color sees 🟨color
         Yes And's graphic ← AndOn's graphic
         No  And's graphic ← AndOff's graphic
  No   And's graphic ← AndOff's graphic
```

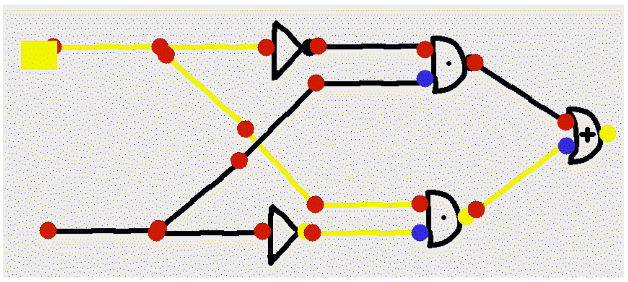


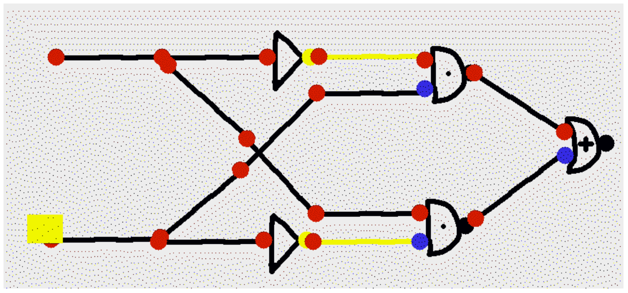


```
Or active ! ticking
Test Or's 🟥color sees 🟨color
  Yes  Or's graphic ← OrOn's graphic
  No   Test Or's 🟩color sees 🟨color
         Yes Or's graphic ← OrOn's graphic
         No  Or's graphic ← OrOff's graphic
```

And, let's make one that will only be off if both the inputs are off. We can see that this is a kind of OR, in that "A OR B" will be "on" if "A is on" or if "B is on" or if they are "both on". This is called an "inclusive OR".

Actually, we only need NOT and AND (or NOT and OR) to make all the possible tests and responses. And an entire computer can be made just from combinations of these two simple "bricks" plus the "mortar" of the wires to hold them together. This is pretty amazing (and also profound)!





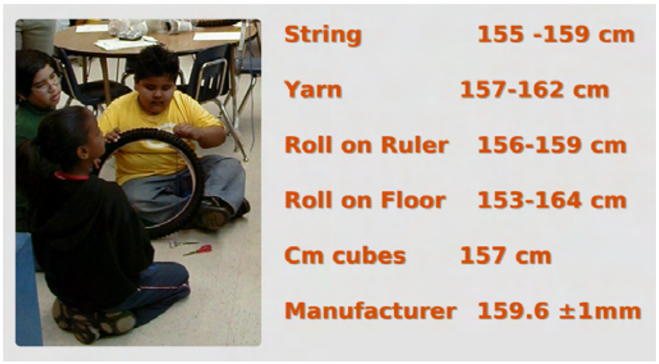
More to come here this summer …

Physical Science: Looking More Carefully At The Outside World

Up to this point our examples have been essentially mathematical, in that they are dealing with computer representations of relationships of ideas. These ideas can be similar to the real world (the models of speed and acceleration), or different from the real world (the car in the speed and acceleration examples was supported by nothing, but didn't fall because there is no "gravity" in the computer world unless we model it). Sometimes we can make up a story that is similar to the real world, and even has a guess that works out. But for most of human history, the guesses about the physical world have been very far from the mark

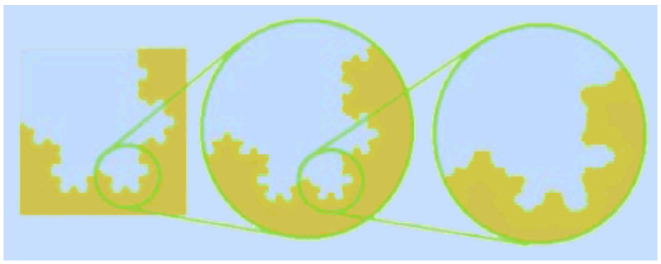
The physical sciences started in earnest when people started to do careful observations and measurements of the physical world, first to do accurate mapping for navigation, exploration and trade, and then to look more closely at more and more phenomena with better instruments and technique.

Another good example of "high-noticing low-cost" is the measuring of the circumference of the bicycle tire project for 5th graders. Much of the philosophical gold in science is to be found in this noticing activity. The students used different materials and got different answers, but were quite sure that there was an exact answer in centimeters (partly because schooling encourages them to get exact rather than real answers).



String	155 -159 cm
Yarn	157-162 cm
Roll on Ruler	156-159 cm
Roll on Floor	153-164 cm
Cm cubes	157 cm
Manufacturer	159.6 ±1mm

What is the circumference of a bike tire?



What is the length of a shore line?

One of the teachers also thought this because on the side of the tire it said it was 20" in diameter. The teacher "knew" that the circumference was pi * diameter, that "pi is 3.14", and "inches times 2.54 converts to 'centimeters' ", etc., and multiplied it out to get the "exact circumference" of the tire = 159.512 cm. We suggested that they measure the diameter and they found it was actually more like 19 and 3/4" (it was uninflated)! This was a shock, since they were all set up to believe pretty much anything that was written down, and the idea of doing an independent test on something written down had not occurred to them.

That led to questions of inflating to different pressures, etc. But still most thought that there was an exact circumference. Then one of us contacted the tire manufacturer (who happened to be Korean) and there were many interesting and entertaining exchanges of email until an engineer was found who wrote back that "We don't actually know the circumference or diameter of the tire. We extrude them and cut them to a length that is 159.6 cm ± 1 millimeter tolerance!"

This really shocked and impressed the children -- the maker of the tire doesn't even know its diameter or circumference! -- and it got them thinking much more powerful thoughts. Maybe you can't measure things exactly. Aren't there "atoms" down there? Don't they jiggle? Aren't atoms made of stuff that jiggles? And so forth. The analogy to "how long is a shoreline?" is a good one. The answer is partly due to the scale and tolerance of measurement. As Mandelbrot and others interested in fractals have shown, the length of a mathematical shoreline can be infinite, and physics shows us that the physical measurement could be "almost" as long (that is very long).

There are many ways to make use of the powerful idea of "tolerance". For example, when the children do their gravity project and come up with a model for what gravity does to objects near the surface of the earth, (see the next project) it is very important for them to realize that they can only measure to within one pixel on their computer screens and that they can also make little slips. A totally literal take on the measurements can cause them to miss seeing that uniform acceleration is what's going on. So they need to be tolerant of very small errors. On the other hand, they need to be quite vigilant about discrepancies that are outside of typical measuring errors. Historically, it was important for Galileo not to be able to measure really accurately how the balls rolled down the inclined plane, and for Newton not to know what the planet Mercury's orbit actually does when looked at closely.

Dropping objects

Children Discover, Measure, and Mathematically Model Galilean Gravity

A nice "real science" example for 11 year olds is to investigate what happens when we drop objects of different weights.

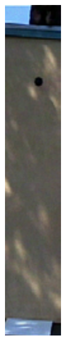
The children think that the heavier weight will fall faster. And they think that a stop-watch will tell them what is going on.

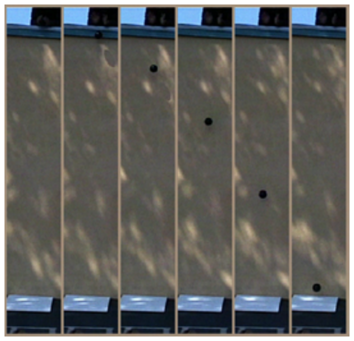
But it is hard to tell when the weight is released, and just when it hits.

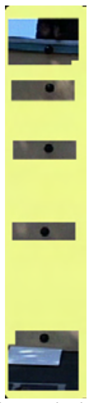
In every class, you'll usually find one "Galileo child". In this class it was a little girl who realized: well, you don't really need the stop watches, just drop the heavy one and the light one and listen to see if they hit at the same time. This was the same insight that Galileo had 400 years ago, and apparently did not occur to any adult, (including the very smart Greeks) for our previous 80,000 years on this planet!

The "Galileo Girl" explains a simple way to see if different weights fall at the same or different speeds.

To really understand in more detail what is going on with gravity near the surface of the Earth, we can use a video camera to catch the dynamics of the dropping weight. We can see the position of the ball frame by frame, 1/30th of a second apart. To make this easier to see we can just pull out every fifth frame and put them side by side:



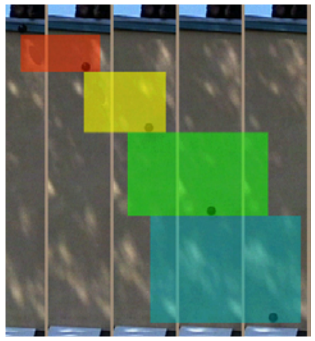


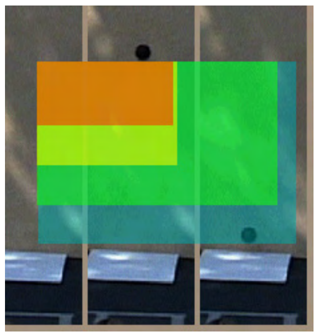

The stacked up frames that reveal the acceleration pattern

Another good thing to do is to take each frame and paint out the non-essential parts and then stack them. When the children do this, most of them will immediately say "Accelleration!" because they recognize that the vertical spacing pattern is the same as the horizontal one they played with using their cars several months before.

But, what kind of acceleration? We need to measure.

Some children will measure directly on the spread frames, while others will prefer to measure the stacked frames.

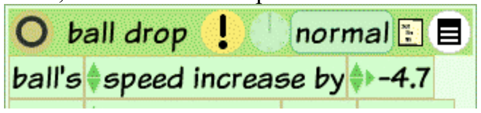




The translucent rectangles help because the bottom of the balls can be seen more accurately, The height of a rectangle measures the speed of the ball at that time (speed is the distance traveled in a unit of time, in this case about 1/5th of a second).

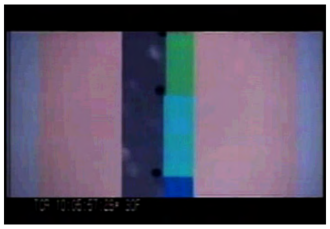
When we stack up the rectangles we can see that the difference in speed is represented by the little strips that are exposed, and the height of each of these strips appears to be the same!

These measurements reveal that the acceleration appears pretty constant, and they made such scripts for their car months ago. Most quickly realize that since the ball is going vertically that they have to write the script so that it is the vertical speed that is increased and the vertical position **y** that is changed. They paint a small round shape to be the simulated ball, and write the script:

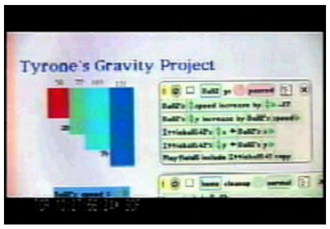


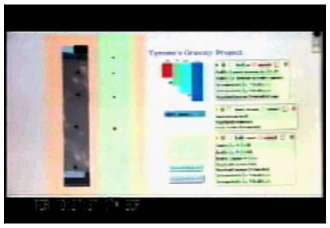
```
ball drop ! normal
ball's speed increase by -4.7
```

Now, how to show that this is a good model for what they have observed? 11 year old Tyrone decided to do as he did with his car months before: to leave a dot copy behind to show that the path of his simulated ball hit the very same positions as the real ball in the video.
Here is what he had to say when explaining what he did and how he did it for the Squeakers video:



> And to make sure that I was doing it just right, I got a magnifier which would help me figure out if I had it - if the size was just right.
>
> After I'd done that I would go and click on the little basic category button and then a little menu would pop up and one of the categories would be Geometry, so I clicked on that.
>
> And here it has many things that have to do with the size and shape of the rectangle. So I would see what the height is… I kept going along the process until I had them all lined up with their height.
>
> I subtracted the smaller one's height from the bigger one to see if there was a kind of pattern anywhere that could help me out. And my best guess worked: so in order to show that it was working, I decided to make – to leave – a dot copy (so that it would show that the ball was going at the exact right speed. And acceleration.)





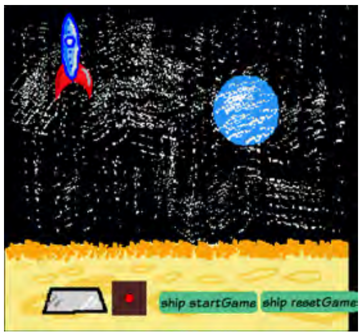
An investigative work of beauty by an 11 year old!

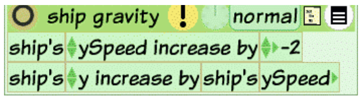
By the way, in the United States about 70 percent of the college kids who are taught about gravity near the surface of the Earth fail to understand it. It's not because the college kids are somehow stupider than the fifth graders, it's because the context and the mathematical approach most college kids are given to learn these ideas are not well suited to the ways they can think.

This subject has been extensively studied with college students in US: 70% (including science majors) fail to understand this Galilean model of gravity near the surface of the Earth. The projects developed in Squeak use a different and simpler kind of incremental mathematics that allows very young children to understand some of the key ideas in calculus. We have found that more than 90% of 5th graders not only understand "Galilean gravity" but are able to derive the mathematical "formulas" (actually 2nd order differential equations) using this alternative mathematics, and author a simulation that matches up very well with the experimental data.

Lunar Lander Game

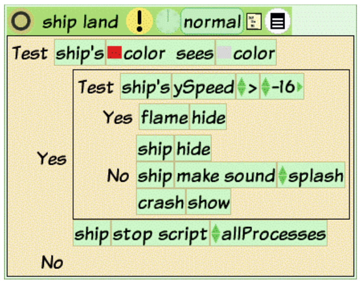
Now that the children have "captured gravity", they can use it to explore other physical situations and to make games. If the ball is repainted into a spaceship, and a moon is made that can be landed on, it is pretty easy for a 12-year old (and most 11 year olds) to make the classic game "Lunar Lander".



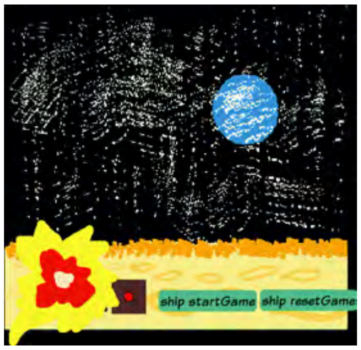


```
ship gravity   normal
ship's ySpeed increase by -2
ship's y increase by ship's ySpeed
```

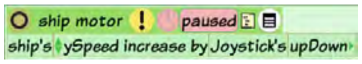
If we just let the gravity script run, it will move the ship downwards (in the negative direction) with more and more negative speed. If it hits the landing plate on the moon's surface while it is moving too fast, the ship should crash. We paint a crash picture and write a script to cause the crash.



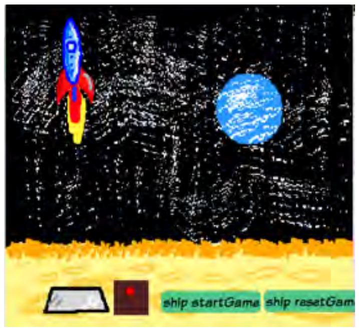
```
ship land   normal
Test ship's 🟥 color sees ◻ color
Yes   Test ship's ySpeed > -16
      Yes flame hide
      No  ship hide
          ship make sound splash
          crash show
      ship stop script allProcesses
No
```



If gravity is a "speed eater" (where what "eats speed" is the downward pointing acceleration **-2**), then we can think of the rocket's motor as a "speed maker" (which imparts a positive upwards acceleration). In test after test, the universe appears to always add such accelerations, and so we can too. We write a script to add in positive acceleration to the rocket's speed. To control this we'll get a joystick from the supplies bin and hook up the updown axis to supply the acceleration.

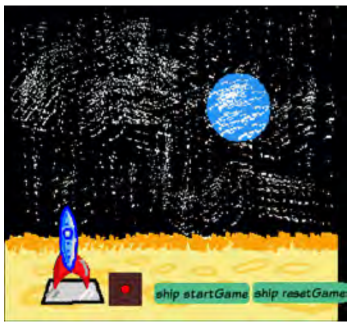


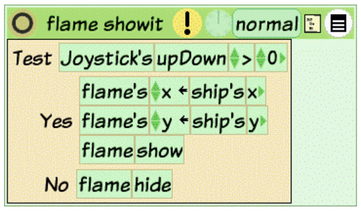
```
ship motor   paused
ship's ySpeed increase by Joystick's upDown
```



Our standard "acceleration imparting change of motion" script doesn't care about how many influences there are on the rocket. Whatever the acceleration is (positive or negative) it will simply increase the speed for each tick. If the speed is negative and the acceleration is positive but smaller, then the ship will still fall (do you see why?). If the acceleration stays positive then the ship's speed will eventually become positive (do you see why?) and the ship will start rising.

It would be nice to see the rocket blast when its motor is running, and this is easy to do. We paint a rocket blast flame and *embed* it into the rocket. We can position the flame so it appears to come from the rocket engine. The flame will now travel with the rocket. (Try it.) Now we can test to see if the joystick updown output is greater than zero. If so, we will show the flame by saying **flame show**. If the joystick updown is zero, we will hide the flame by saying **flame hide**.





```
flame showit   normal
Test Joystick's upDown > 0
Yes  flame's x ← ship's x
     flame's y ← ship's y
     flame show
No   flame hide
```

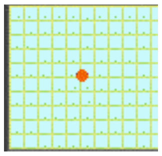
Spaceships And Inertia

We can use the idea of inertia to come up with how a physical body (like a spaceship) will behave in space.

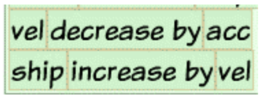
The nifty thing here is that the very same two-line "acceleration to velocity to change of motion" script will work unchanged – *this is because vectors are a way to represent numbers that are independent of a coordinate system and rotations*. This, and because vectors can be represented visually by decomposable arrows that are very intuitive for young children, is why teaching numbers as vectors is a very strong pedagogical strategy.

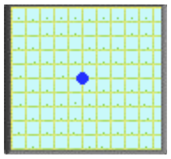
The only new thing we need to think about is how change of motion is done when the spaceship is moving in one direction, *but oriented in a different direction*, and fires its motors.

Each Etoy object is secretly a vector, so we just have to paint a red dot for acceleration and a blue one for velocity, and put them in a playfield object (this is like a little page that holds things and also is a kind of separate space or universe for the objects.

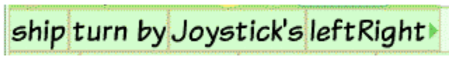

Making a vector for acceleration.

We tell the playfields to orient to the center and now the location of the colored dots is also their vector value. The change of motion parts of the **move** script are exactly what we've been using for all of our accelerated motion simulations

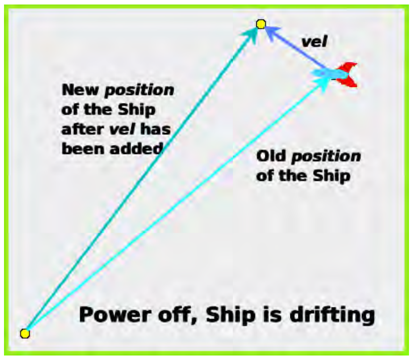

```
vel decrease by acc
ship increase by vel
```


... and another vector for velocity

As in the Lunar Lander project, we will use the joystick to control the spaceship's engine (by the updown direction), but we now will want to use the leftRight direction to turn the ship.

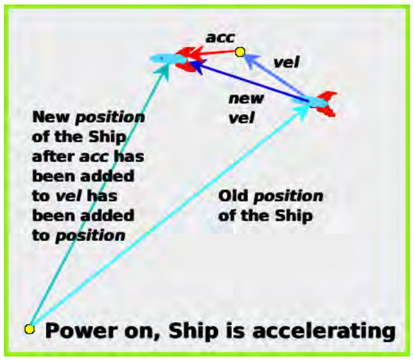

```
ship turn by Joystick's leftRight
```

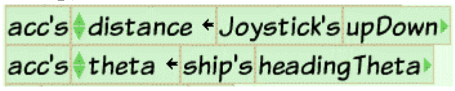
Of course, there is rotational inertia, so this is too simple, since the ship will quit rotating as soon as the leftRight output of the joystick is zero. However, we can use this because modern day spaceships act on this control just this way: they use a combination of steering jets, gyroscopes, and computers to allow the pilots to simply steer the orientation of their vehicle.


vel
New position of the Ship after vel has been added
Old position of the Ship
Power off, Ship is drifting

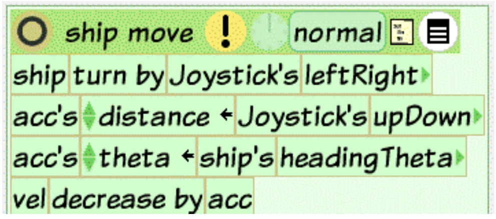
Now we need to deal with the very common case that the ship will be moving in one direction and oriented in another, and we then fire the engines. The diagram for this looks like:

We can see that the size (**distance**) of the new acceleration vector is gotten from the how the motor affects the ship – so we can just use **joystick's upDown** for this. Now we need to set the direction (**theta**) of the acceleration vector, and we can take this from the ship's own **heading** (because the motor is aimed in the same direction as the ship).


acc
vel
new vel
New position of the Ship after acc has been added to vel has been added to position
Old position of the Ship
Power on, Ship is accelerating

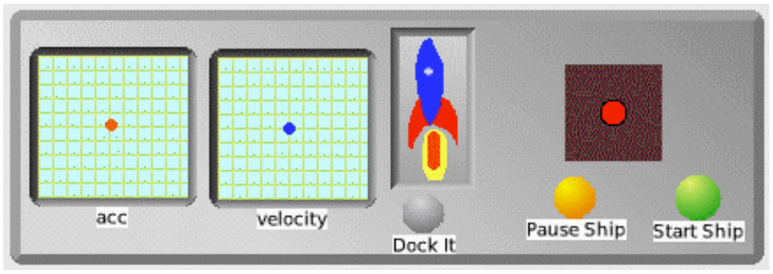

```
acc's distance ← Joystick's upDown
acc's theta ← ship's headingTheta
```

Putting all of these together gives us the following very simple (and very pretty) script.

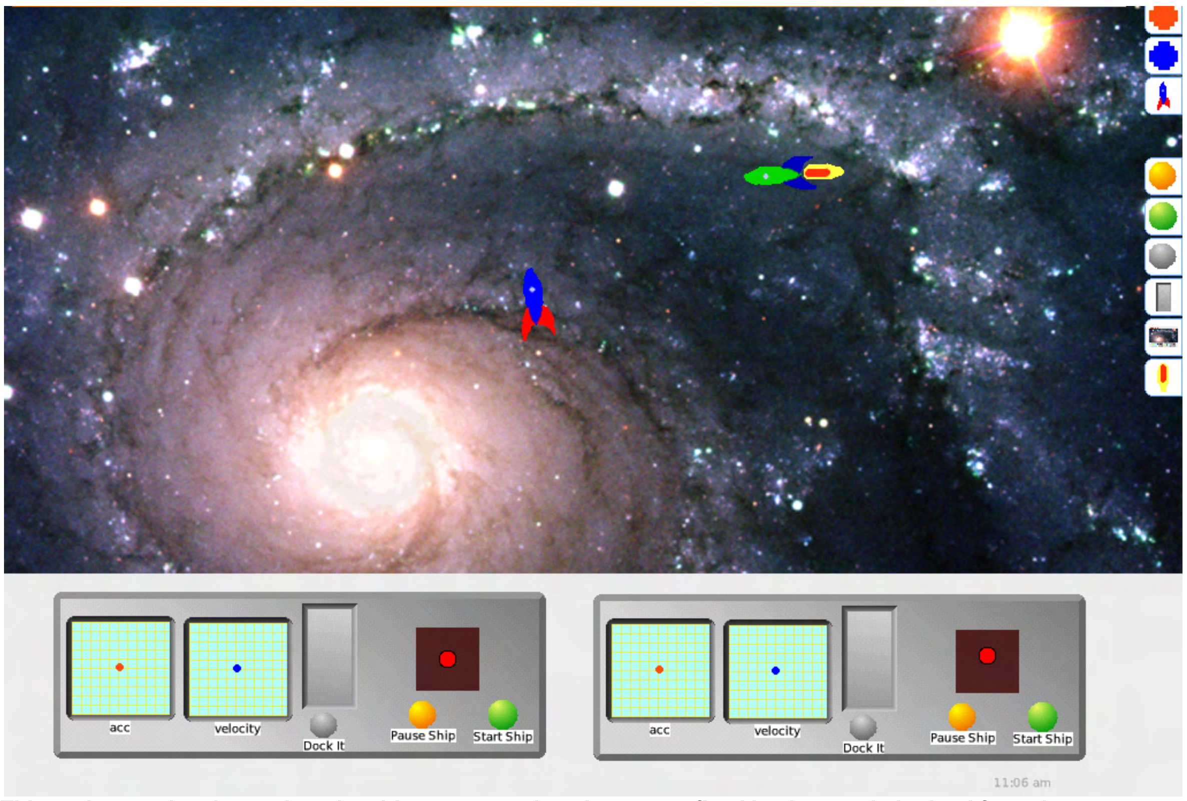

```
ship move ! normal
ship turn by Joystick's leftRight
acc's distance ← Joystick's upDown
acc's theta ← ship's headingTheta
vel decrease by acc
```

This works nicely! Now we can add some pleasant details. We can use the same logic for the flame as we did in Lunar Lander.

Now we can package all of this up in a nice "control panel" (using a playfield to hold everything). We add buttons to pause the ship, start the ship (move it from the dock to the universe it will fly in), and to bring the ship back to its dock. This panel looks like:



Playfields have another nice property. As we mentioned, they are like a little private universe for holding things – and they also remember all the relationships that objects have with each other. So, if we copy a playfield, we will get a copy of all the relation-ships. In this case, we get a new spaceship, vel, acc, and controls. We can repaint this in a different color, and now two people can play this game!

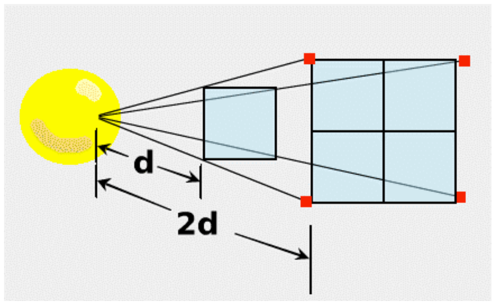


This project works nicely when the ships are placed against a star-filled background obtained from the Internet.

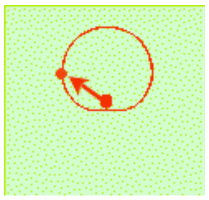
Newtonian Orbits

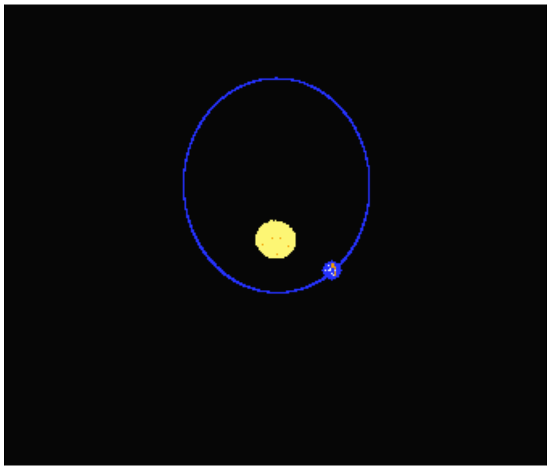
This is a difficult project to do "scientifically" because it is not easy to directly measure gravity as an "inverse square of distance" force in "child's scale" on the surface of the Earth. The children are able to get a "pretty much constant" acceleration from their movie of balls dropping about 14 feet from the top of their school building. This is Galileo's result also. This is because the difference in force due to distance from the top of the building to the ground is about 1 part in one million!

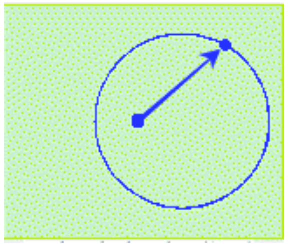
However, children can do a mathematical exploration that is based on a reasonable premise derived from an analogy to the way light disperses from a glowing body.



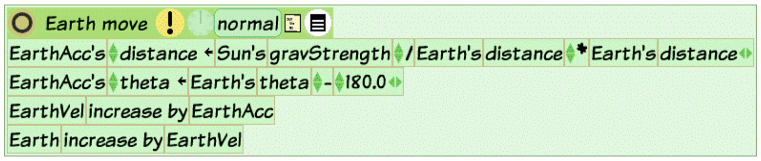
We can directly measure the intensity of light from a glowing body, and we discover that it acts as though light is emitted in straight-line "rays" that spread out from the surface. This means that the "number" of rays passing through an area at a distance d will have to pass through 4 times that area at double the distance (2d) and thus the intensity of the rays is ¼. We can say that "the intensity of light is inversely proportional to the square of the distance". What if we try the idea that gravity might also be emitted in a similar way?


Earth's acceleration vector over time magnified by 1000

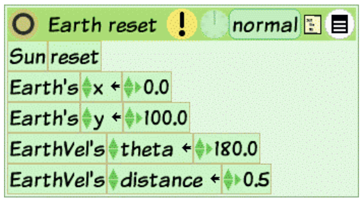



Earth's velocity vector over time magnified by 50. Notice that it is a displace circle. This is not easy to prove!

The simulated Earth in orbit around the simulated Sun. The images of the Earth's acceleration, velocity and position were all captured in the same instant. We can see from the velocity that the Earth is slowing down as it moves farther away from the Sun, and we can see from the acceleration that it is getting much smaller.



```
Earth move  normal
EarthAcc's distance ← Sun's gravStrength / Earth's distance * Earth's distance
EarthAcc's theta ← Earth's theta - 180.0
EarthVel increase by EarthAcc
Earth increase by EarthVel
```



```
Earth reset  normal
Sun reset
Earth's x ← 0.0
Earth's y ← 100.0
EarthVel's theta ← 180.0
EarthVel's distance ← 0.5
```

The initial conditions for this orbit

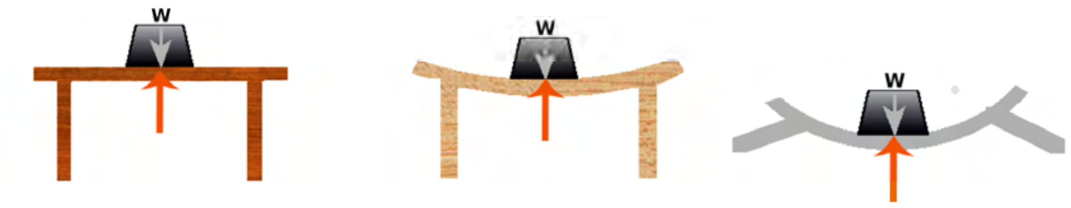
We make up the magnitude (called distance in Etoys) of the acceleration vector from the gravStrength of the Sun scaled by the inverse square of the distance. The direction (called theta) in Etoys of this vector is the same as the Earth but pointing at the Sun. From here it's just "increase-by" in 2D. This is why "numbers as vectors" is so nice!

Lots Of Things Are Springy!
One of the problems with classical science is that it reused common words – like know, theory, force, etc. – for very new ideas. New words should have been chosen. For example, for most children and adults, the word "force" is a very confusing term for something that only influences a change of motion.

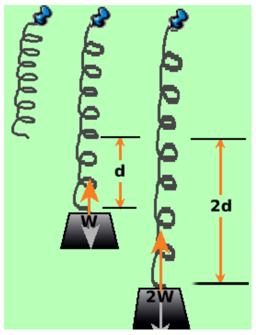
In Science, the weight of an object is thought of as a "force" in the direction of the source of gravity. But if we place a weight on a sturdy table so that it moves no further, most people do not think of the table exerting an upwards force that balances the downward force of the weight.

If we try a table made from a flimsy wood, like balsa, we will see the table bend until it either breaks or is able to exert a balancing upwards force.

If we try this on a paper table, it will simply collapse down to the floor which exerts the upwards force.



And, if we watch carefully, we can see that the balsa wood table will move up and down when the weight is placed on it, as though it were a kind of spring (which it is).
Beams have the same characteristics. And all these are examples of "springy things".



If we look at a spring with a weight hung on it we can measure how much it stretches until it is able to balance the force.

If we put on twice the weight most springs will stretch very close to twice as much. This gives us a way to simply describe the force of the spring: *it is proportional to the length it is stretched*.

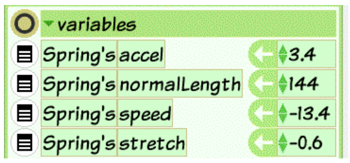
Now we can use the ideas about acceleration and speed we learned from the falling weight. But the acceleration is not constant now because it is proportional to the stretch of the spring. What can we do?

What is really nice here is that we can get the computer to compute very small little movements in which we can pretend the acceleration is constant. Then we can measure the stretch of the spring and do this again. This gives us a very simple but good model of the spring, and is a great example of how easy it is for young children to learn the ideas of calculus with the environment of a computer.

We simply paint a spring-shaped object and look in its geometry category. We find a property called **Spring's length** which is the vertical size of our spring picture. We are measuring the stretch of the spring so we need to make a new property called **Spring's normalLength** to remember this.

Our accelerated movement mathematics is:

```
Spring's speed increase by Spring's accel
Spring's length increase by Spring's speed
```



```
variables
Spring's accel            3.4
Spring's normalLength     144
Spring's speed            -13.4
Spring's stretch          -0.6
```

There are two influences on the acceleration. The weight on the spring is always going to be accelerated downwards by gravity regardless of whatever else is going on (this is like the ball) so we can just pick a constant number for this: let's say 4. And, moment by moment, there will be another acceleration exerted by how much the spring is stretched at that point. We can get the stretch by subtracting the current length from the normal length:

Spring's stretch <- Spring's normalLength – Spring's length

We need to scale this down so it will act like a regular spring. Scaling to 5% will work well.

Spring's stretch multiply by 0.05

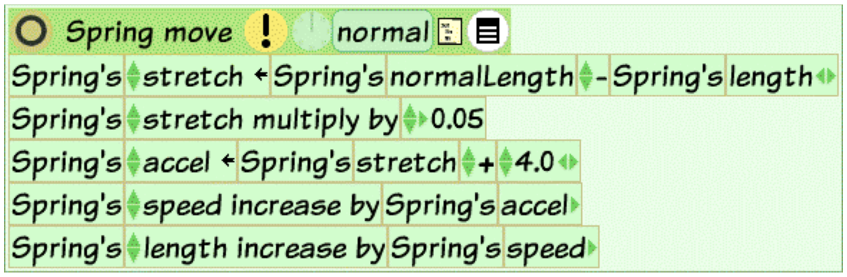
Now we can get the total acceleration by adding the two "forces" together.

Spring's accel <- Spring's stretch + 4

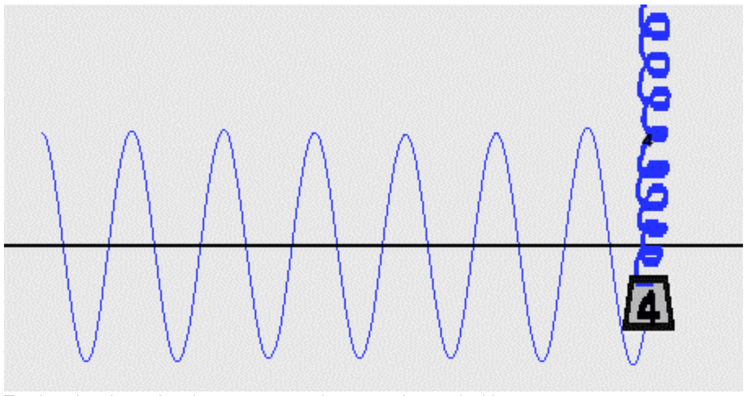
And we can calculate the movement for this short time tick using our normal math:

Spring's speed increase by Spring's accel
Spring's length increase by Spring's speed

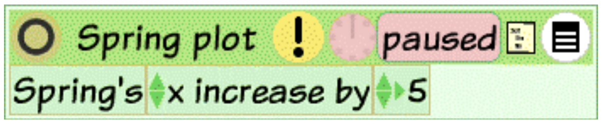
Our thought process can be put directly into the spring's movement script:



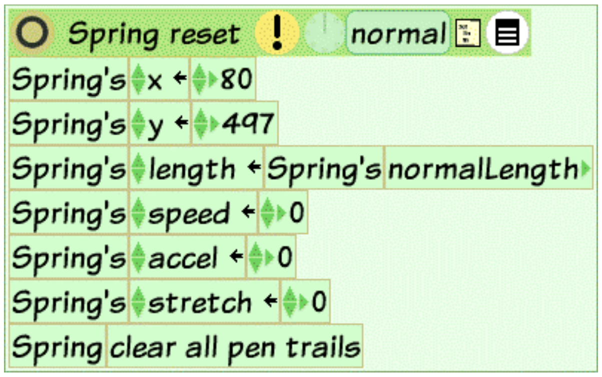
```
Spring move   normal
Spring's stretch ← Spring's normalLength - Spring's length
Spring's stretch multiply by 0.05
Spring's accel ← Spring's stretch + 4.0
Spring's speed increase by Spring's accel
Spring's length increase by Spring's speed
```



To plot what the spring does, we turn on its pen and move it sideways



```
Spring plot   paused
Spring's x increase by 5
```

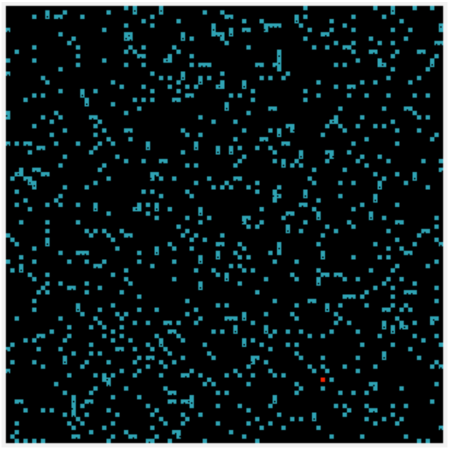


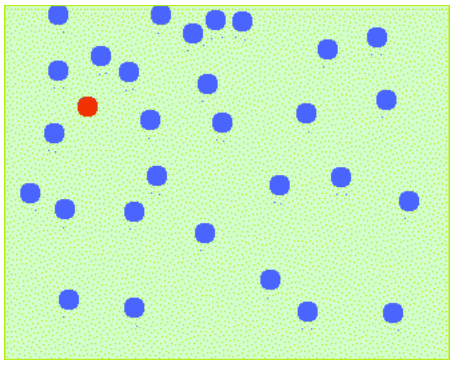
```
Spring reset   normal
Spring's x ← 80
Spring's y ← 497
Spring's length ← Spring's normalLength
Spring's speed ← 0
Spring's accel ← 0
Spring's stretch ← 0
Spring clear all pen trails
```

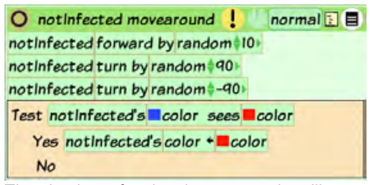
The Beauty And Importance Of Complex Systems

Etoys can make many thousands of copies of objects and their behaviors and rund them all at once. Because of this we can explore very complex systems by just scripting the behavior of one item and making many copies of it.

For example, if we make lots of little dots, we can explore the behavior of contagious processes, such as rumors and disease. Here the scripts are very simple, and cause a dot to change color when it collides with an "infected" dot.


Just "one little dot". One infected person with an infectious incurable disease (the little red dot in the lower right corner) in the general population of 1000 people will eventually cause everyone to become infected.


We just make a blue dot as a "non infected" villager and create as many copies as we want, then make the last one a red infected one. The green rectangle is their "village"

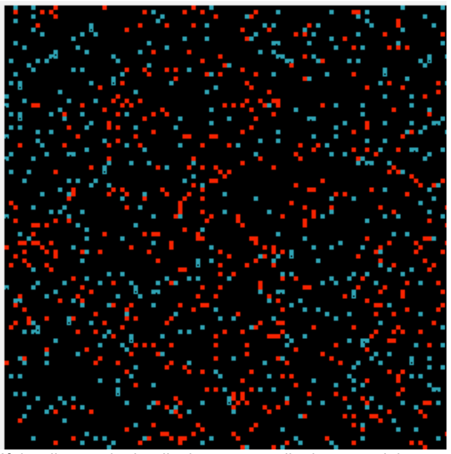


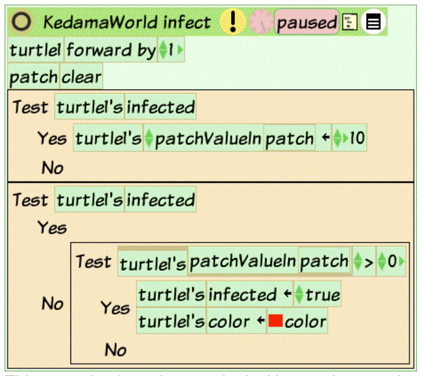
```
notInfected moveAround ! normal
notInfected forward by random 10
notInfected turn by random 90
notInfected turn by random -90
Test notInfected's 🟦color sees 🟥color
    Yes notInfected's color ← 🟥color
    No
```

The simplest of scripts just moves the villagers around and tests to see if there is a contact with an infected person. If so, the villager becomes infected (the color is changed to red).

The size of the arena for collisions determines the delays between collisions, and allows us to explore matters of life and death, such as really understanding the characteristics of epidemics: fast deadly ones like typhoid which are very noticeable, and slow deadly ones such as AIDS (which is deadly in part because the onset of an AIDS epidemic is not dramatic).

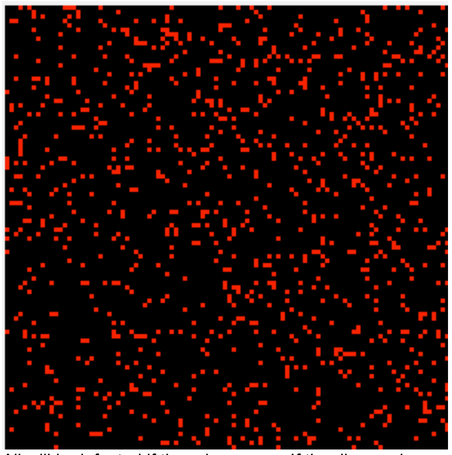
A poor understanding of slow deadly epidemics in many parts of the world is one of the main causes of the AIDS disaster. People have to reach beyond their common sense into the "uncommon sense" of models for disasters in order to help their imagination spur them to early action. The computer will eventually create an even larger change in how humans think about ideas than the printing press.


If the disease is deadly then eventually there won't be enough health people to do anything about the infection. If the incubation period is long, then the population might not notice in time.



```
KedamaWorld infect ! paused
turtle1 forward by 1
patch clear
Test turtle1's infected
    Yes turtle1's patchValueIn patch ← 10
    No
Test turtle1's infected
    Yes
    No  Test turtle1's patchValueIn patch > 0
            Yes turtle1's infected ← true
                turtle1's color ← 🟥color
            No
```

This very simple script can deal with any size population. It controls the bigger village shown on the right.

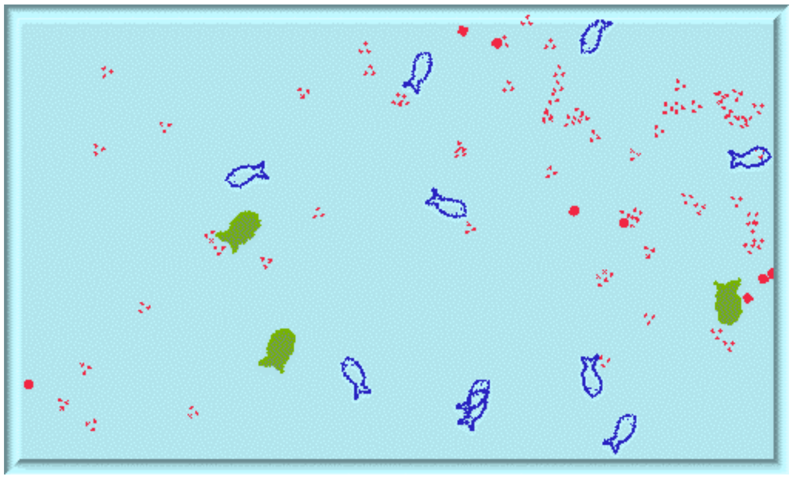

All will be infected if there is no cure. If the disease is deadly then all will eventually die. Human brains are not good at dealing with slow disasters. We need simulations to help us think about important things.

A Major Project: An Aquarium Ecology

The Vivarium Project was one of our large multiyear (7 years) investigations with elementary aged children. The Vivarium's vehicle was ecologies of various kinds (including human ecologies) and the project was aimed at better understanding of how to help children learn about systems, model them, and think in terms of complex systems.

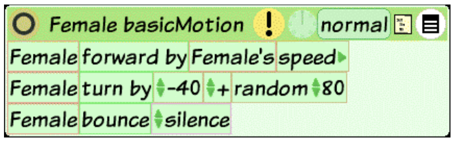
Since animals need to eat other living entities in order to survive, there are a very large number of situations in which a balance is achieved between "eaters" and "the eaten" – and this often occurs as a chain of such dependencies.

Here is a project that is simple by ecological standards – but requires the children to do a lot of thinking and modeling to make it: fish and plankton in an aquarium.



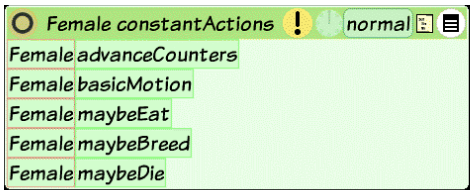
Here we see female fish (with blue boundaries), male fish (green), plankton (the larger red dots), and plankton "residue" (plankton that has been eaten).

Both the plankton and the fish swim randomly about, and the children are easily able to come up with this behavior, (which is so pervasive that it is found everywhere in the animal kingdom (and in much smaller particles thanks to the kinetic motion of heat and collisions between them).



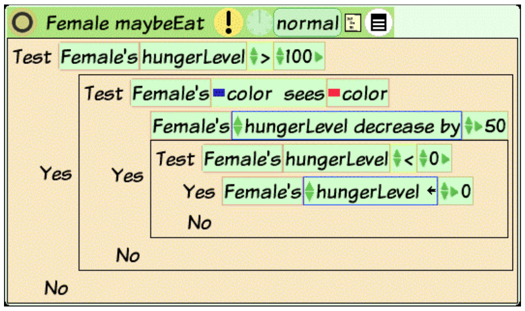
```
Female basicMotion   normal
Female forward by Female's speed
Female turn by -40 + random 80
Female bounce silence
```

The fish are motivated by their nervous system to try do a number of things at once: to move around, to eat, to mate, to get old etc. We can make a script that "fires off impulses" to do the various actions:

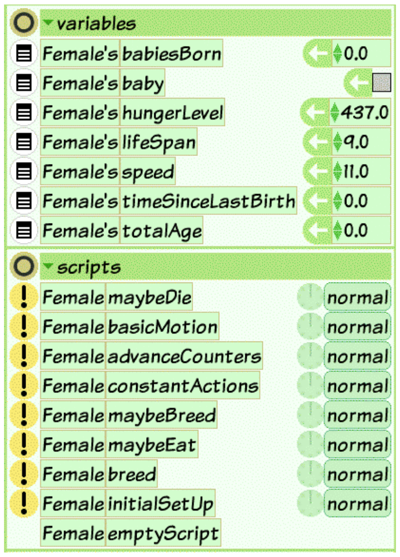


```
Female constantActions   normal
Female advanceCounters
Female basicMotion
Female maybeEat
Female maybeBreed
Female maybeDie
```

Decisions are constantly being made by the fish's simulated nervous system. Here is the "maybe eat?" script:



```
Female maybeEat   normal
Test Female's hungerLevel > 100
Yes
    Test Female's color sees color
    Yes
        Female's hungerLevel decrease by 50
        Test Female's hungerLevel < 0
        Yes Female's hungerLevel ← 0
        No
    No
No
```



variables

Female's babiesBorn	0.0
Female's baby	
Female's hungerLevel	437.0
Female's lifeSpan	9.0
Female's speed	11.0
Female's timeSinceLastBirth	0.0
Female's totalAge	0.0

scripts

Female maybeDie	normal
Female basicMotion	normal
Female advanceCounters	normal
Female constantActions	normal
Female maybeBreed	normal
Female maybeEat	normal
Female breed	normal
Female initialSetUp	normal
Female emptyScript	

Looking At (and Smelling) The Outside World

Here is a project that is "not quite science, but suggestive", and is part of a larger set of experiments to start plausible thinking about the very small. We ask the children to bring "really smelly stuff" from home – perfume, ammonia, (hopefully not stuff that should go down the toilet), etc. We close all the windows of the classroom (or go to the auditorium if there is one) and ask the children to raise their hands when they can smell something. (This is best done first with a few bottles that have nothing smelly in them, to get the children to relax and not try to blindly please the teacher.)

If there is no air moving in the room, a really smelly substance opened at one end of the room will first be smelled by the children closest to it, and we should see a spreading wave of hands going up. This should open questions and discussions about what is going on, and ways to explain it.

Children in a room with smelly stuff

Reacting to the smelly stuff

A second experiment can be done with fans to create wind. One ploy could be to have some students stand on the other side of the door to the room, open the door, and see if they can smell the substance. Then children in the room can use big hand fans to try to move the air in the room out. The students outside should quickly detect a lot more smell and feel "wind".

Fans

Reactions

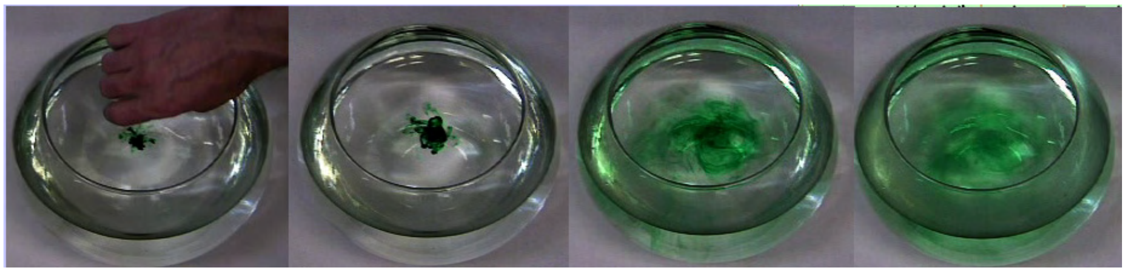
A third experiment can be done with water and sugar each filling half of equal sized containers. We put both of them on the scales to see what they weigh. We ask them to think about what might happen to both the volume and the weight if we combine the sugar and the water. Most of the children will decide that the total weight should be the same, and the volume should double

With the water still on the scales, we pour the sugar into the water and put its container back on the scales. We'd like the children to notice and realize that the weight of the containers, sugar and water is still the same. But to their great surprise, the combined volume is not double the single volumes but a little less.

Putting sugar in the water container

The volume is less than the sum, but the weight stays the same!

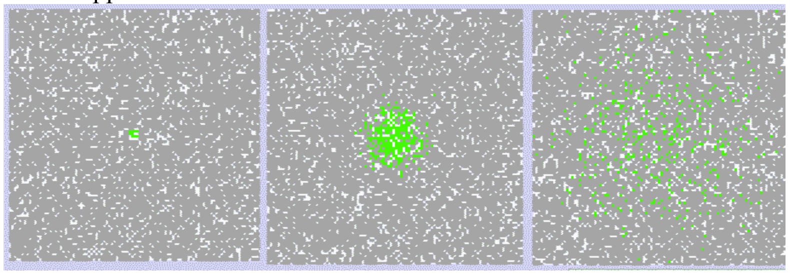
A fourth experiment can be done with a little food dye, a clear bowl containing water, and the camera on the XO. We drop a little food dye into the bowl, watch what happens, and also capture the experience with the camera. We can play back the sequence to see what is happening. And we can also play back five minutes at 60 times real-time to see if that provides more insight.

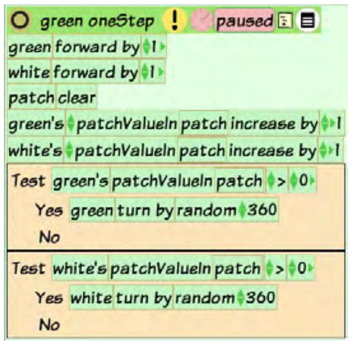


One theory that covers all four experiments is that when you divide substances finely enough, you get individual particles that allow things to fit in between. This is "suggestive" but isn't yet science. We can make it more suggestive (but still not "science") by making a computer model.

Etoys can have hundreds and even thousands of objects all dynamically behaving the same way. We can make a "room" or a "bowl" by using particles of one color (let's say 2000 gray ones) to represent one of the substances. And we can use another color (say 500 green ones) for the new substance to be introduced. In all cases, somehow the perfume or dye particles (if indeed made out of particles) moved and dispersed. One guess would be that the particles are already moving around randomly and we are just seeing (or smelling) the movement.

We just made a single Etoys object do a random walk. We can use this directly to move the particles and see what happens.





```
green oneStep  !  paused
green forward by 1
white forward by 1
patch clear
green's patchValueIn patch increase by 1
white's patchValueIn patch increase by 1
Test green's patchValueIn patch > 0
    Yes green turn by random 360
    No
Test white's patchValueIn patch > 0
    Yes white turn by random 360
    No
```

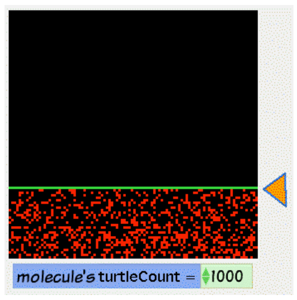
It looks a lot like the bowl of water and dye. But this is not yet what science would call a *theory* (the official scientific term is *conjecture*). In this case we can't see what is actually "down at the bottom" and whether or not it is particles or continuous or something quite different is beyond our ability to observe. In cases like these (and most of science is this way), we have to gather a lot of evidence from many different cases to make the conjecture plausible enough to eventually be judged a real theory.

Only 120 years ago, matter as made up from particles was still controversial amongst the top scientists in the world, and it was only 100 years ago that enough evidence and good enough models were gathered to convince the scientific world that matter can be subdivided into particles. (In fact, shortly thereafter, in the 1920s, the quantum conjectures started to become real theories, and these have evidence and mathematics showing particles of matter at smaller scales have some of the properties of radiation.)

Today, it's not quite clear what is going on at subatomic scales, but at the level of the matter we most easily perceive – solids, liquids, gases – we can safely act as though these are made from particles.

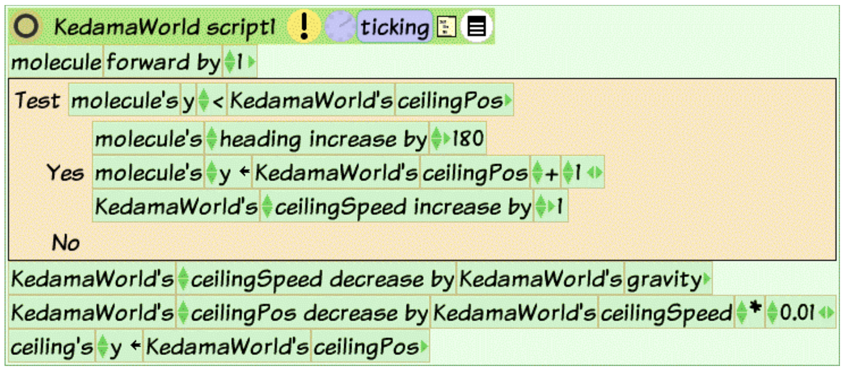
The theory that most of our sensible matter is made from tiny particles in motion is too important to "just tell the children" as though it is a religious fact. Even though it is very difficult to lay the basis for the particle theory (as it indeed was for scientists 100 years ago), every effort has to be made to make these conclusions as truly scientific as possible. We will take this up again further on in this summary when we talk about chemistry.

Piston Supported By A Gas

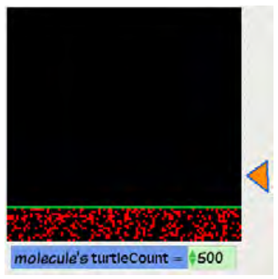


What if we take our particle diffusion model that just moves particles around randomly and, instead of confining the particles in a box, we let the top of the box be a piston that has a weight on it. If we shoot a small marble at a big one, the larger one will only move a little. If we shoot a bigger marble, then the big one will move more. So, we can take as a reasonable guess that each time a small particle hits the piston it will impart just a little acceleration upwards (like the rocket motor but really tiny) and meanwhile gravity will be dragging down the weighted piston.

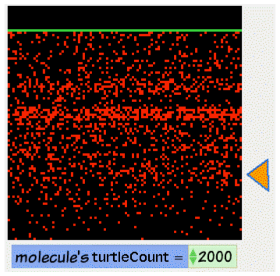
Gravity will be exerting a downwards pull on the little particles as well, but we will assume they are so energetic that this has no noticeable effect. “Kedama” is the Japanese word for “lots of mess”, so it is a good name for this chaotic system!



```
KedamaWorld script1   ticking
molecule forward by 1
Test molecule's y < KedamaWorld's ceilingPos
        molecule's heading increase by 180
    Yes molecule's y ← KedamaWorld's ceilingPos + 1
        KedamaWorld's ceilingSpeed increase by 1
    No
KedamaWorld's ceilingSpeed decrease by KedamaWorld's gravity
KedamaWorld's ceilingPos decrease by KedamaWorld's ceilingSpeed * 0.01
ceiling's y ← KedamaWorld's ceilingPos
```

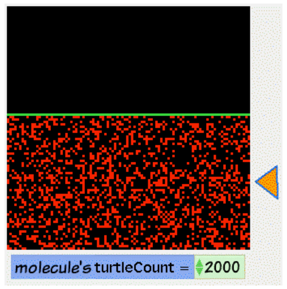


The crux of this simulation is that enough particles are hitting the piston on average per a short duration of time to impart a “pretty continuous” upwards acceleration on the piston. The downwards acceleration from the weight of the piston will try to move the piston downwards. The further down the piston moves, the higher the probability that more particles will hit it. At some point, if the weight is not too heavy, the downwards and upwards accelerations will “pretty much balance” and we just see small fluctuations in the level of the piston.



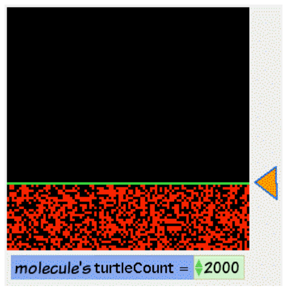
Now we can do some experiments. For example, we can mark the level of the piston as it is balanced by 1000 particles. Then we can reduce the number of particles to 500 and see what happens. The piston goes to a level that is about half the height (and this makes some sense because we have reduced the probability of collisions between the particles and the piston by 50%.

Now let’s suddenly put in 2000 total particles. We get an explosion! As the piston settles back we see shock waves as bunches of particles bounce back and forth. Eventually, we get a result we should have anticipated: the piston gets balanced at about twice the height it had with 1000 particles and 4 times that with 500 particles.



We can also change the weight of the piston. What do you think will happen if we double the weight of the piston?

These are mathematical experiments, but they provide a lot of insight for how to set up the classic Boyle’s Law experiments in the physical world.



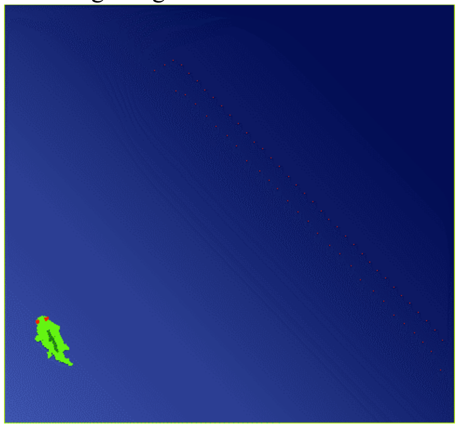
Following Chemical Signals

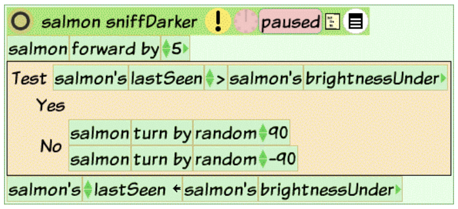
This project is in two parts. First, we model typical animal behavior that is used to follow chemical signals in the environment by being able to sense the relative concentration of the chemical and to be able to remember a past smell and concentration well enough to decide to keep going or to try a different path.

We'll pick a salmon swimming upstream to lay its eggs in the same part of the river in which it was born.

In our simulation we'll avoid the dramatic leaping backwards up the waterfalls, and concentrate on how the salmon might be able to guide itself by smelling a particular chemical from its spawning ground and being able to remember just the concentration of the last sniff it took.

To model the water with the chemical in it, we'll use a gradient of color, where the darker the color, the more concentrated is the chemical. Etoys lets us sense not just the color under an object, but also the brightness. So, for this simulation, less brightness means "getting closer".



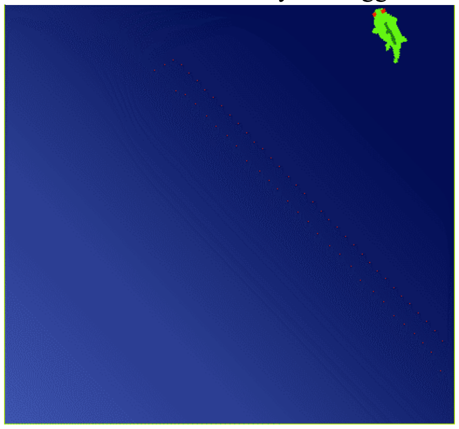


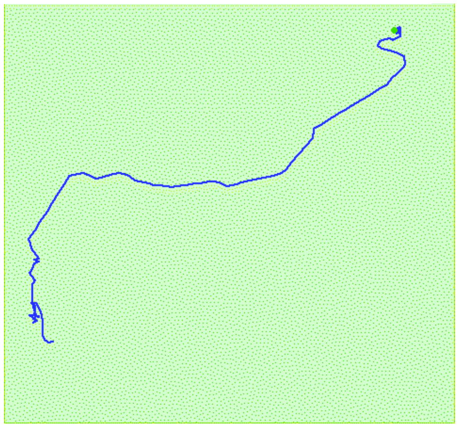
```
salmon sniffDarker ! paused
salmon forward by 5
Test salmon's lastSeen > salmon's brightnessUnder
  Yes
  No   salmon turn by random 90
       salmon turn by random -90
salmon's lastSeen ← salmon's brightnessUnder
```

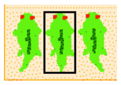
This is the classic strategy used in most animals from bacteria on up to make progress with incomplete information from the environment.

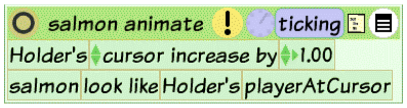
1: *keep moving*
2: *if things aren't better, do something random*
3: *remember last information from environment.*

Below we see the salmon has successfully found the darkest corner, and to the right we see the path it took. The script under the path is a "turtle" following the salmon's position and drawing a trail in a different playfield. The holder and script under the "river" animate the salmon's body to wriggle as a fish does.

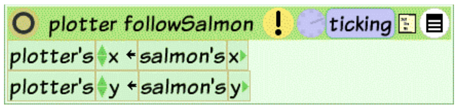






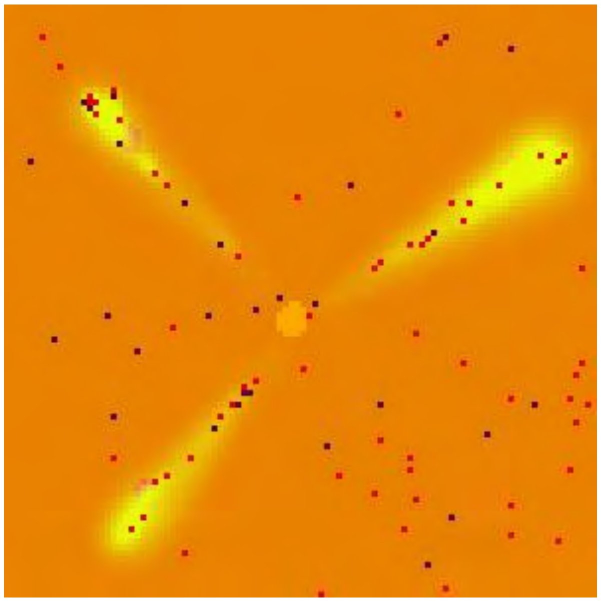


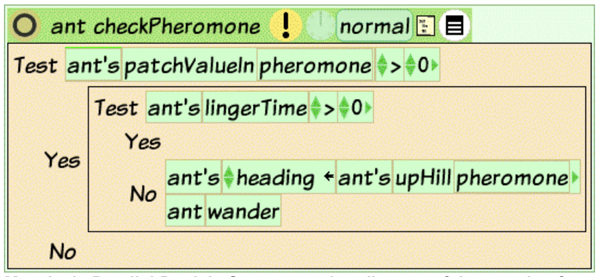
```
salmon animate ! ticking
Holder's cursor increase by 1.00
salmon look like Holder's playerAtCursor
```



```
plotter followSalmon ! ticking
plotter's x ← salmon's x
plotter's y ← salmon's y
```

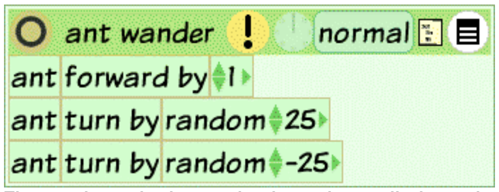
Ants use their ability to sense and follow gradients to communicate with each other: they lay scent trails to mark paths to help other ants find “interesting things” (usually food). Ants are a kind of “swarm animal” and often act like a larger organism whose “cells” can independently sense, think and do.


Massively Parallel Particle System can handle tens of thousands of moving and background particles. This is a simulation of ants gathering food,and leaving a diffusing trail of pheromones to guide other ants.



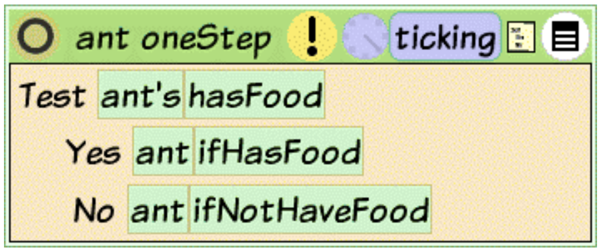
```
ant checkPheromone  !  normal
Test ant's patchValueIn pheromone > 0
Yes
    Test ant's lingerTime > 0
    Yes
    No  ant's heading ← ant's upHill pheromone
        ant wander
No
```

Massively Parallel Particle System can handle tens of thousands of moving and background particles. This is a simulation of ants gathering food,and leaving a diffusing trail of pheromones to guide other ants.



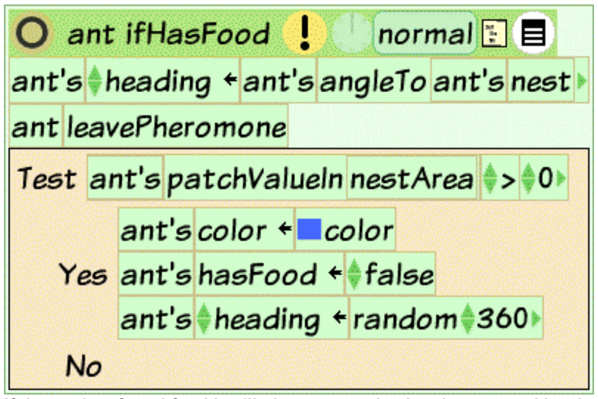
```
ant wander  !  normal
ant forward by 1
ant turn by random 25
ant turn by random -25
```

The wander script is our classic random walk that we’ve used for so many kinds of investigations.



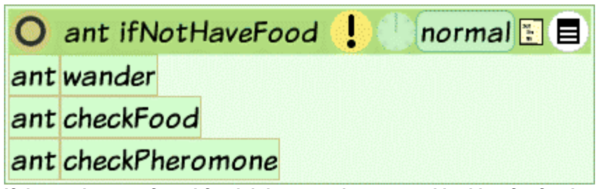
```
ant oneStep  !  ticking
Test ant's hasFood
    Yes ant ifHasFood
    No  ant ifNotHaveFood
```

This is the “master” script that fires off all the rest of the scripts for each ant. It is very simple.



```
ant ifHasFood  !  normal
ant's heading ← ant's angleTo ant's nest
ant leavePheromone
Test ant's patchValueIn nestArea > 0
Yes ant's color ← 🟦 color
    ant's hasFood ← false
    ant's heading ← random 360
No
```

If the ant has found food it will plot a course back to its nest and level a pheromone trail of diffusing evaporating odor. The diffusion is done similarly to the dye in the goldfish bowl.



```
ant ifNotHaveFood  !  normal
ant wander
ant checkFood
ant checkPheromone
```

If the ant has not found food, it just wanders around looking for food or for a pheromone trail left by other ants. Scientists are pretty sure that most food is found when the randomly stumbles into it.

Media Authoring In Etoys

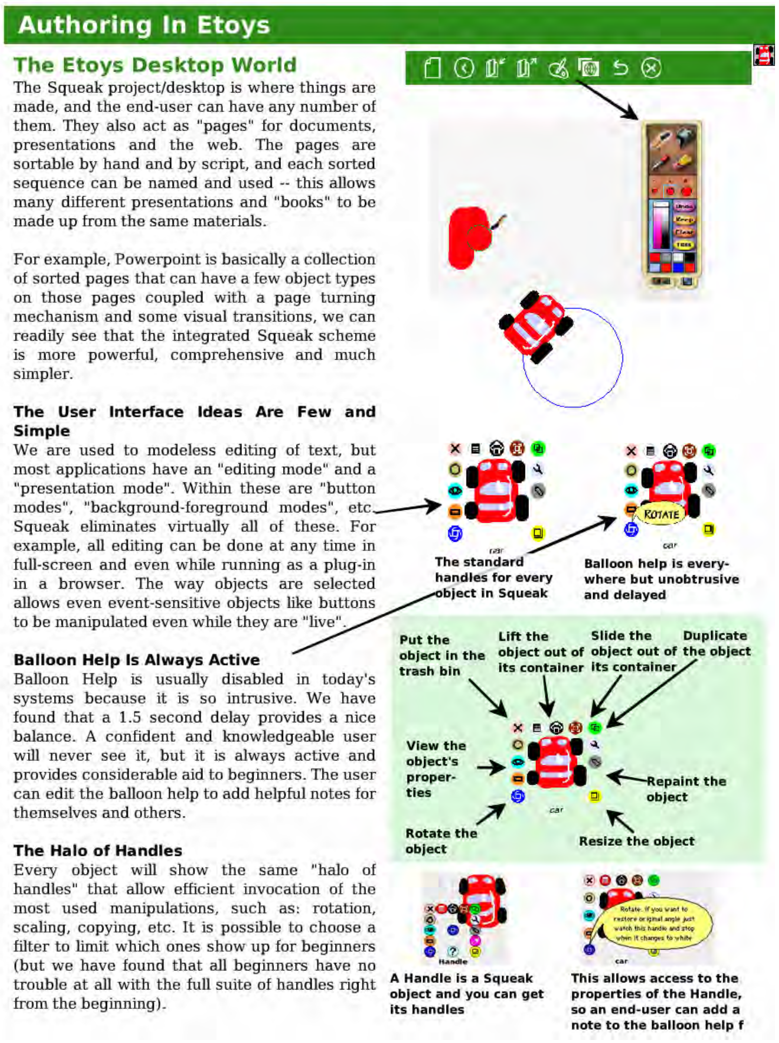
Etoys was also inspired by the invention of Desktop Publishing at Xerox PARC and by the subsequent very nice approach to end-user authoring in Hypercard.

The picture to the right is direct output from an Etoy project (on an OLPC display screen) of the next page in this document (done in Microsoft Word). The OLPC display has very high resolution – so the details are crisp – but has a small size – basically like a paperback book (4.5" by 6"), rather than an 8.5" by 11" sheet of paper. This means that we have to be careful of the visual angle especially of the font sizes when developing for the OLPC XO.

To be able to develop anywhere, Etoys has a high quality display zoom that allows any sized window (including 4.5" by 6" OLPC sized windows) to be used. The developer can develop in any size that is convenient and can move back and forth if, for example, both a large screen and small screen layout is the goal.

All Development Can Be Done on the OLPC

A very important point, to be made again, is that the Etoys system is complete and includes all of its development tools in its small footprint on the OLPC machine. Any child, and parent, any teacher, any curriculum developer can develop at all levels (including the underlying powerful Squeak Smalltalk base) on the OLPC XO.



Authoring In Etoys

The Etoys Desktop World

The Squeak project/desktop is where things are made, and the end-user can have any number of them. They also act as "pages" for documents, presentations and the web. The pages are sortable by hand and by script, and each sorted sequence can be named and used -- this allows many different presentations and "books" to be made up from the same materials.

For example, Powerpoint is basically a collection of sorted pages that can have a few object types on those pages coupled with a page turning mechanism and some visual transitions, we can readily see that the integrated Squeak scheme is more powerful, comprehensive and much simpler.

The User Interface Ideas Are Few and Simple

We are used to modeless editing of text, but most applications have an "editing mode" and a "presentation mode". Within these are "button modes", "background-foreground modes", etc. Squeak eliminates virtually all of these. For example, all editing can be done at any time in full-screen and even while running as a plug-in in a browser. The way objects are selected allows even event-sensitive objects like buttons to be manipulated even while they are "live".

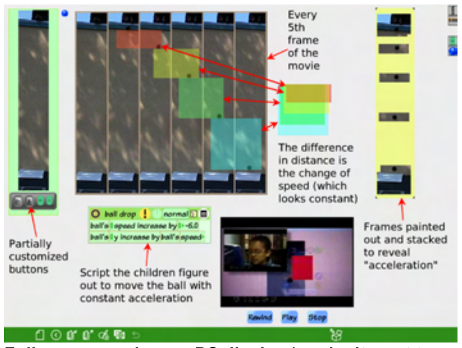
Balloon Help Is Always Active

Balloon Help is usually disabled in today's systems because it is so intrusive. We have found that a 1.5 second delay provides a nice balance. A confident and knowledgeable user will never see it, but it is always active and provides considerable aid to beginners. The user can edit the balloon help to add helpful notes for themselves and others.

The Halo of Handles

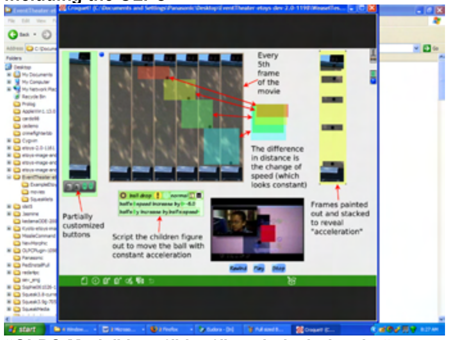
Every object will show the same "halo of handles" that allow efficient invocation of the most used manipulations, such as: rotation, scaling, copying, etc. It is possible to choose a filter to limit which ones show up for beginners (but we have found that all beginners have no trouble at all with the full suite of handles right from the beginning).

car
The standard handles for every object in Squeak
ROTATE
car
Balloon help is everywhere but unobtrusive and delayed
Put the object in the trash bin
Lift the object out of its container
Slide the object out of its container
Duplicate the object
View the object's properties
Repaint the object
car
Rotate the object
Resize the object
Handle
A Handle is a Squeak object and you can get its handles
Rotate. If you want to restore original angle just watch this handle and stop when it changes to white
car
This allows access to the properties of the Handle, so an end-user can add a note to the balloon help f

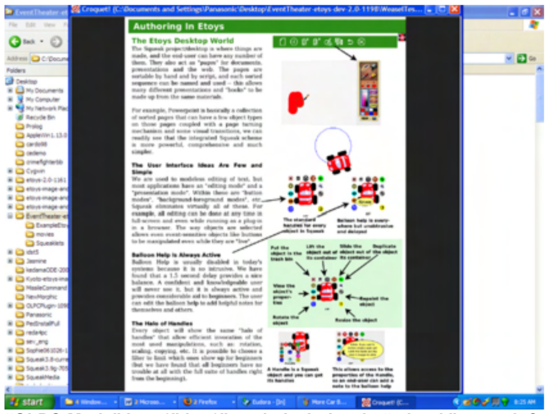


Every 5th frame of the movie
The difference in distance is the change of speed (which looks constant)
Partially customized buttons
ball drop
normal
ball's speed increase by -6.0
ball's y increase by ball's speed
Script the children figure out to move the ball with constant acceleration
Frames painted out and stacked to reveal "acceleration"
Rewind
Play
Stop

Full screen mode on a PC display (we don't want to see MS Windows and do wish to use the entire display. This is the way Etoys is usually used on other platforms, including the OLPC



"OLPC Mode" in a 6" by 6" scaled window in "presentation" landscape format on a PC display.



"OLPC Mode" in a 6" by 6" scaled window in "e-book" portrait format on a PC display.

Media Authoring In Etoys

The Etoys Desktop World

The Squeak project/desktop is where things are made, and the end-user can have any number of them. They also act as "pages" for documents, presentations and the web. The pages are sortable by hand and by script, and each sorted sequence can be named and used – this allows many different presentations and "books" to be made up from the same materials.

For example, Powerpoint is basically a collection of sorted pages that can have a few object types on those pages coupled with a page turning mechanism and some visual transitions, we can readily see that the integrated Squeak scheme is more powerful, comprehensive and much simpler.

The User Interface Ideas Are Few and Simple

We are used to modeless editing of text, but most applications have an "editing mode" and a "presentation mode". Within these are "button modes", "background-foreground modes", etc. Squeak eliminates virtually all of these. For example, all editing can be done at any time in full-screen and even while running as a plug-in in a browser. The way objects are selected allows even event-sensitive objects like buttons to be manipulated even while they are "live".

Balloon Help Is Always Active

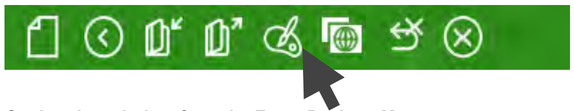
Balloon Help is usually disabled in today's systems because it is so intrusive. We have found that a 1.5 second delay provides a nice balance. A confident and knowledgeable user will never see it, but it is always active and provides considerable aid to beginners. The user can edit the balloon help to add helpful notes for themselves and others.

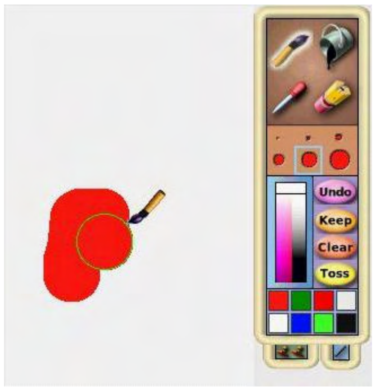
The Halo of Handles

Every object will show the same "halo of handles" that allow efficient invocation of the most used manipulations, such as: rotation, scaling, copying, etc. It is possible to choose a filter to limit which ones show up for beginners (but we have found that all beginners have no trouble at all with the full suite of handles right from the beginning).

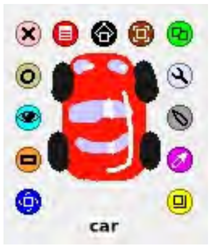
The Handles Are Also Squeak Etoys Objects

(Everything is.) This means that end-users have the option of customizing everything in the system. Most of the time this will not be done, but e.g. for young children, a teacher may wish to add more extensive notes to the existing balloon help, or make the handles appear quicker. Squeak has "fences" that warn about changes and entering more complex territories, but the end-users can still have the option to explore.

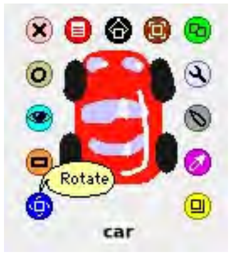

Getting the paintbox from the Etoys Desktop Menu



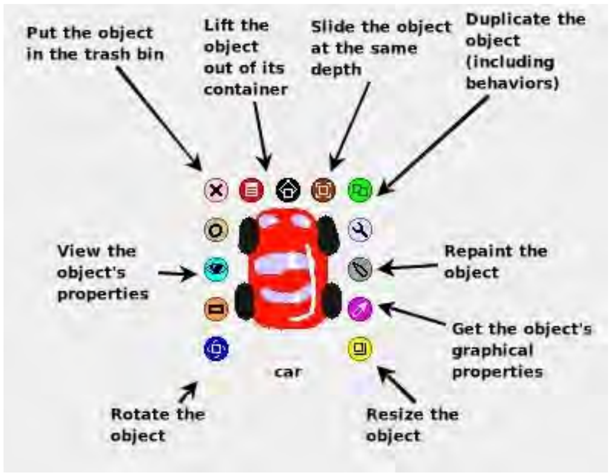
Painting a car on the desktop

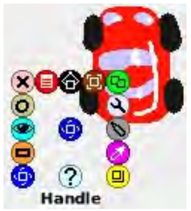


The standard handles for every object in Squeak

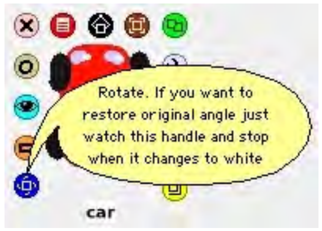


Balloon help is everywhere but unobtrusive and delayed


Put the object in the trash bin
Lift the object out of its container
Slide the object at the same depth
Duplicate the object (including behaviors)
View the object's properties
Repaint the object
Get the object's graphical properties
car
Rotate the object
Resize the object
What some of the handles do



A Handle is a Squeak object and you can get its handles


Rotate. If you want to restore original angle just watch this handle and stop when it changes to white
car
This allows access to the properties of the Handle, so an end-user can add a note to the balloon help for additional aid

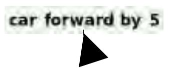
Objects Are "The Same" But May Wear Different Costumes
We've just seen that all objects will show the same control handles, but the similarities between them go much deeper. Each object's visible appearance is a costume that it wears (and can change), but underneath the objects are the same. (This is very much like the actors in a play. They have different roles, look different on the stage – they may even be playing a tree – but underneath they are all human beings, with all the similar properties that humans share.)

Objects Reveal Their Properties and Behaviors the Same Way

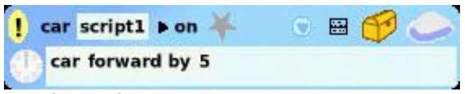
Clicking on the viewer handle will bring up the object's viewer, which shows all of the object's properties and behaviors organized into categories. Every object is "the same": is graphical, can carry other objects, can be scripted, has a pen, etc., so most of each viewer is exactly the same from object to object.

The Car Painting and Car Viewer Are Both Costumes For Car
And the end-user can make more for any object.

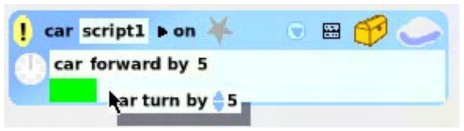
All Objects Are Scripted The Same Way
Scripts are made by dragging out tiles onto the desktop, and then dragging tiles into the script. Syntax is always correct.


car forward by 5

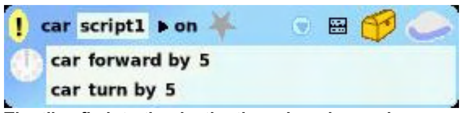
Tiles dragged from the viewer and dropped on the desktop ...


car script1 ▸ on
car forward by 5

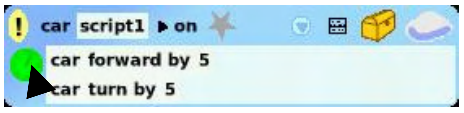
... make a script


car script1 ▸ on
car forward by 5
car turn by 5

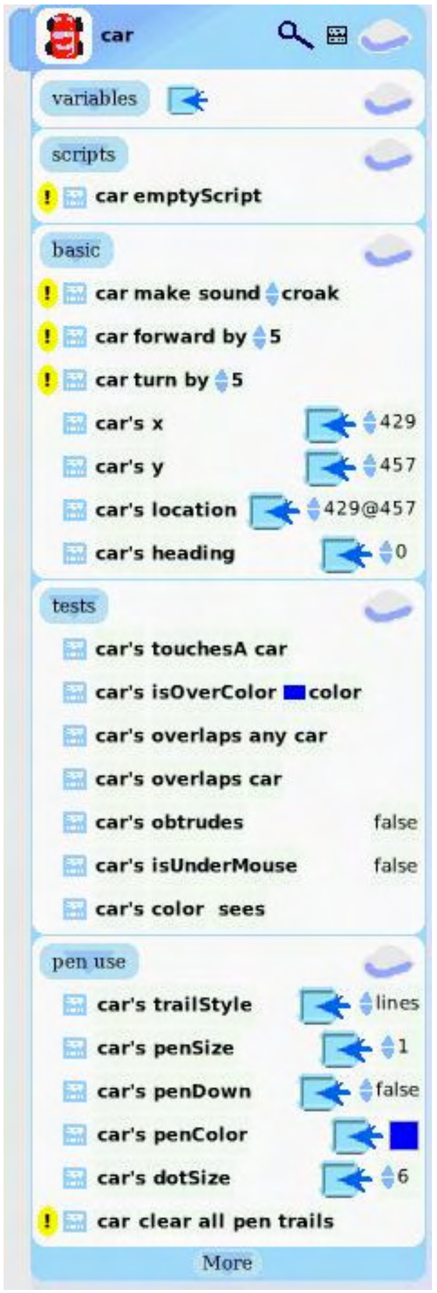
The green "destination marker" shows up when tiles are dragged over a script.


car script1 ▸ on
car forward by 5
car turn by 5

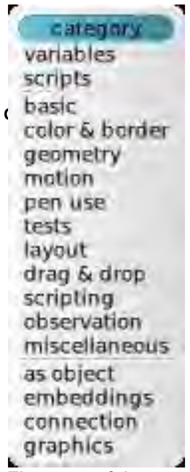
The tiles fly into the destination when dropped.


car script1 ▸ on
car forward by 5
car turn by 5

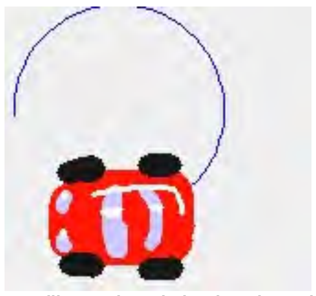
The script can be started "ticking" by clicking on the clock


car
variables
scripts
car emptyScript
basic
car make sound croak
car forward by 5
car turn by 5
car's x 429
car's y 457
car's location 429@457
car's heading 0
tests
car's touchesA car
car's isOverColor color
car's overlaps any car
car's overlaps car
car's obtrudes false
car's isUnderMouse false
car's color sees
pen use
car's trailStyle lines
car's penSize 1
car's penDown false
car's penColor
car's dotSize 6
car clear all pen trails
More

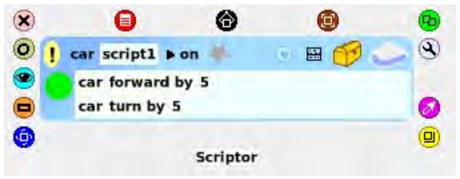
Object viewer with some of the standard categories o
properties and behaviors


category
variables
scripts
basic
color & border
geometry
motion
pen use
tests
layout
drag & drop
scripting
observation
miscellaneous
as object
embeddings
connection
graphics

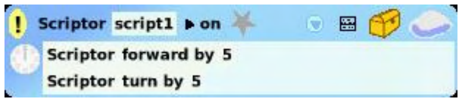
The menu of the standard categories



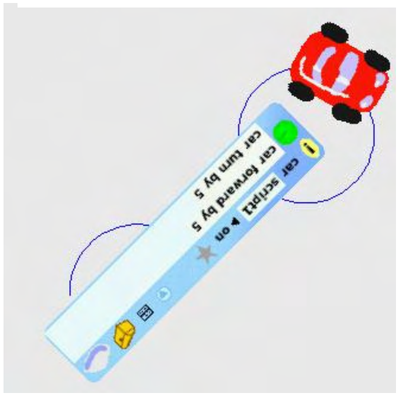
The car will move in a circle when the script ticks. If we have made "pen down" (in the "pen use" category) true, then the car will leave a trace as it moves.



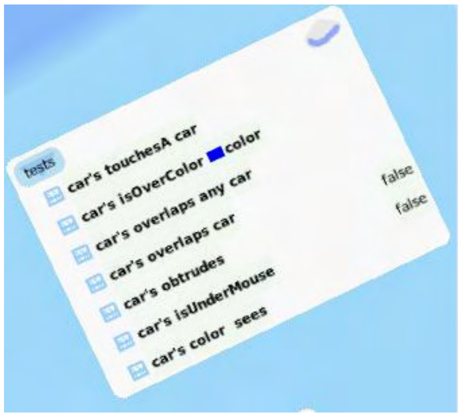
Because the script is a standard Squeak Etoys object, we can get its handles, and its viewer …



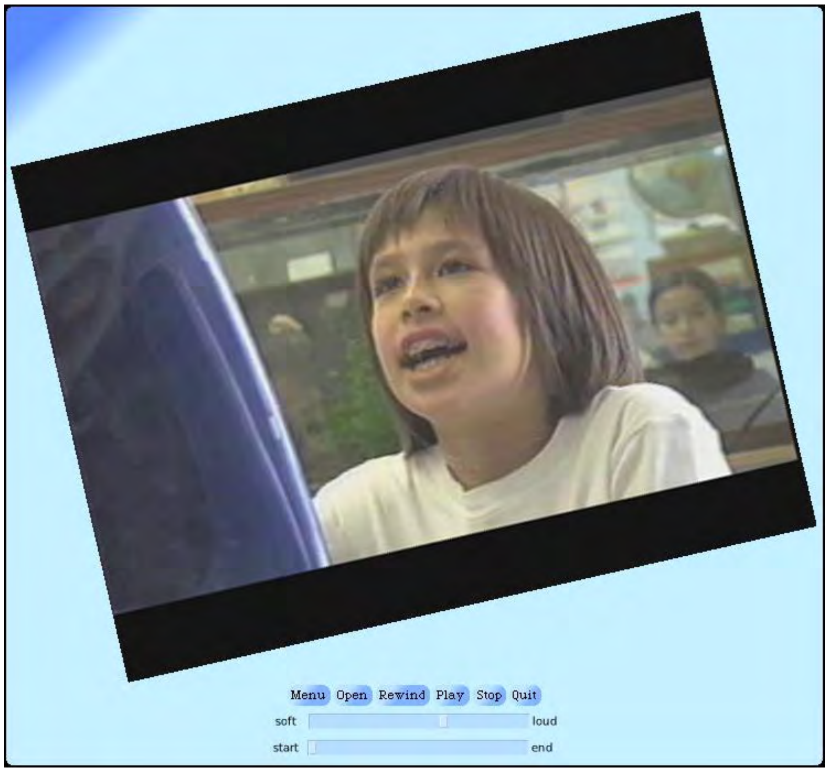
… and write the same script we wrote for the car



The result is that the scriptor will move in a circle and leave a pen trail of its own

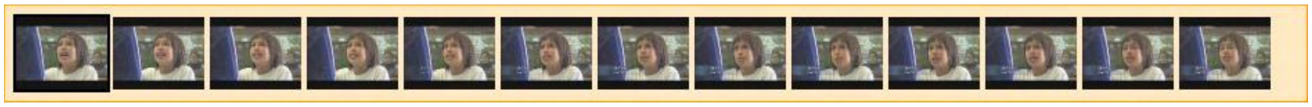


We should guess correctly that we can do the same thing for any Squeak object, including a category in a viewer!



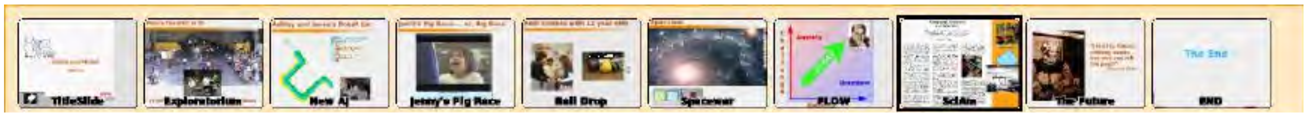
Or a movie even while it is playing!

Movies and Videos Are Just "Animations"
Movies are represented (and abstracted) in Etoys as Animations.

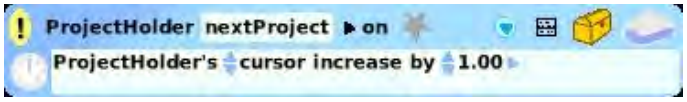


What's missing? Movies usually have hundreds to thousands of frames, each of which has lots of pixels. These frames are usually held as a file on the hard drive and are brought into the player application. Squeak Etoys provides automatic services for relating contents of files to Etoy objects, and also to compress and decompress pictures.

"Books" and Presentations Are Just "Animations"
Squeak "Book" (multipage documents that are like a Hypercard stack) and Squeak Presentations (like powerpoint but more flexible) are the very same kind of structure. The "pages" can be any object (including a whole project),

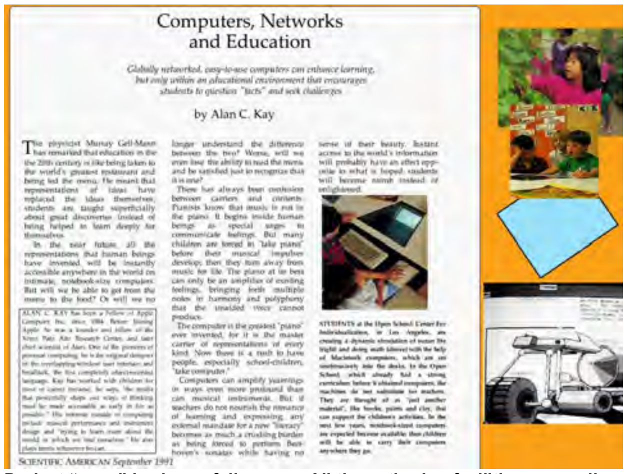


and turning is done by a script that looks like:



```
ProjectHolder nextProject ▸ on
  ProjectHolder's cursor increase by 1.00
```

We will take a deeper look at how media is manipulated and made in subsequent "Media Examples".



Computers, Networks and Education

Globally networked, easy-to-use computers can enhance learning, but only within an educational environment that encourages students to question "facts" and seek challenges

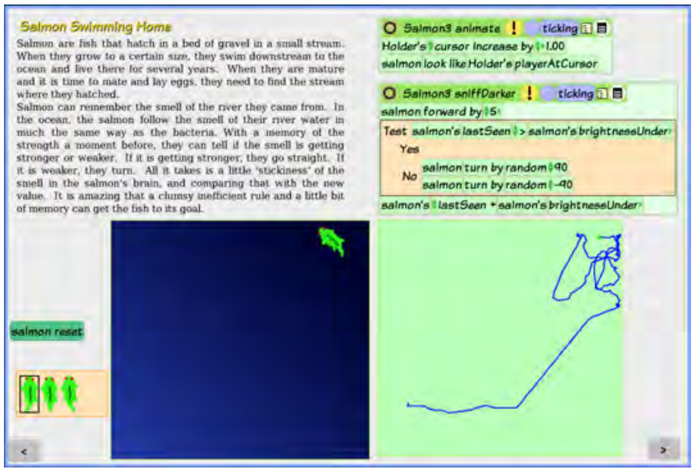
by Alan C. Kay

[illegible]

SCIENTIFIC AMERICAN September 1991

Project "page" is shown full screen. All the authoring facilities are available, this is a live project, not just an image.

Here are examples of an "Active Essay" about the powers of agitation, testing, and just a little memory when things are going well or badly. We use this as part of the lead up to the Theory of Evolution to show that small amounts of memory coupled with sensing can change "astronomically improbable" processes into "pretty darn probable" ones.



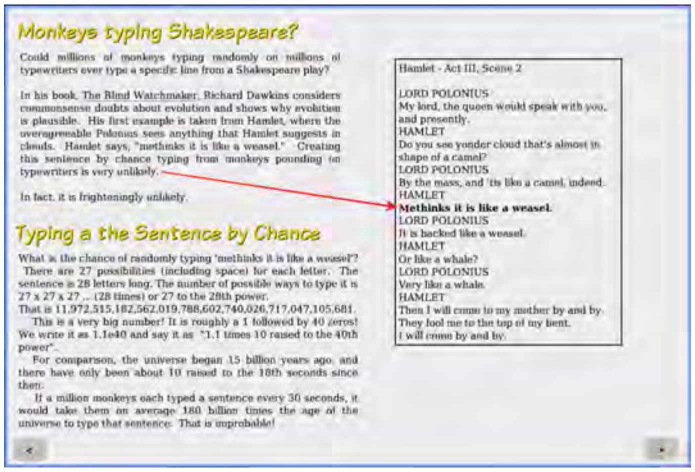
Salmon Swimming Home

Salmon are fish that hatch in a bed of gravel in a small stream. When they grow to a certain size, they swim downstream to the ocean and live there for several years. When they are mature and it is time to mate and lay eggs, they need to find the stream where they hatched.
Salmon can remember the smell of the river they came from. In the ocean, the salmon follow the smell of their river water in much the same way as the bacteria. With a memory of the strength a moment before, they can tell if the smell is getting stronger or weaker. If it is getting stronger, they go straight. If it is weaker, they turn. All it takes is a little 'stickiness' of the smell in the salmon's brain, and comparing that with the new value. It is amazing that a clumsy inefficient rule and a little bit of memory can get the fish to its goal.

```
Salmon3 animate ! ticking
Holder's cursor increase by 1.00
salmon look like Holder's playerAtCursor

Salmon3 sniffDarker ! ticking
salmon forward by 5
Test salmon's lastSeen > salmon's brightnessUnder
  Yes
  No  salmon turn by random 90
      salmon turn by random -90
salmon's lastSeen ← salmon's brightnessUnder
```

One of the early pages in the essay is our old friend the salmon and its swim upstream, but now included as a dynamic example in this longer explanation.



Monkeys typing Shakespeare?

Could millions of monkeys typing randomly on millions of typewriters ever type a specific line from a Shakespeare play?

In his book, The Blind Watchmaker, Richard Dawkins considers commonsense doubts about evolution and shows why evolution is plausible. His first example is taken from Hamlet, where the overagreeable Polonius sees anything that Hamlet suggests in clouds. Hamlet says, "methinks it is like a weasel." Creating this sentence by chance typing from monkeys pounding on typewriters is very unlikely.

In fact, it is frighteningly unlikely.

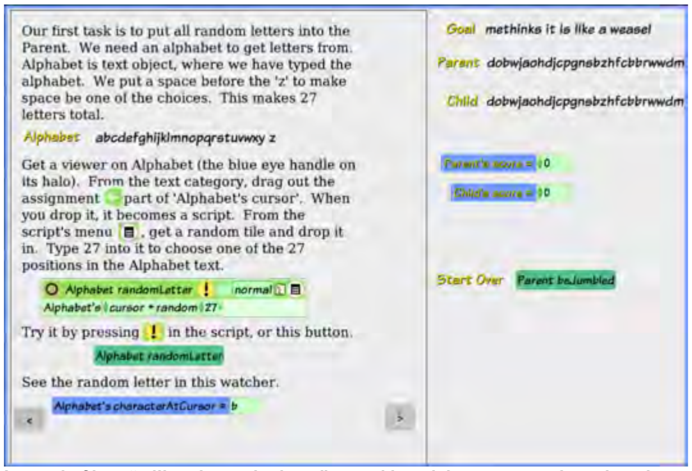
Typing a the Sentence by Chance

What is the chance of randomly typing 'methinks it is like a weasel'? There are 27 possibilities (including space) for each letter. The sentence is 28 letters long. The number of possible ways to type it is 27 x 27 x 27 ... (28 times) or 27 to the 28th power.
That is 11,972,515,182,562,019,788,602,740,026,717,047,105,681.
This is a very big number! It is roughly a 1 followed by 40 zeros! We write it as 1.1e40 and say it as "1.1 times 10 raised to the 40th power".
For comparison, the universe began 15 billion years ago, and there have only been about 10 raised to the 18th seconds since then.
If a million monkeys each typed a sentence every 30 seconds, it would take them on average 180 billion times the age of the universe to type that sentence. That is improbable!

Hamlet - Act III, Scene 2

LORD POLONIUS
My lord, the queen would speak with you, and presently.
HAMLET
Do you see yonder cloud that's almost in shape of a camel?
LORD POLONIUS
By the mass, and 'tis like a camel, indeed.
HAMLET
Methinks it is like a weasel.
LORD POLONIUS
It is backed like a weasel.
HAMLET
Or like a whale?
LORD POLONIUS
Very like a whale.
HAMLET
Then I will come to my mother by and by.
They fool me to the top of my bent.
I will come by and by.

Richard Dawkins used the improbability of zillions of monkeys typing by chance to illustrate that many people think most processes are totally random.



Our first task is to put all random letters into the Parent. We need an alphabet to get letters from. Alphabet is text object, where we have typed the alphabet. We put a space before the 'z' to make space be one of the choices. This makes 27 letters total.

Alphabet abcdefghijklmnopqrstuvwxy z

Get a viewer on Alphabet (the blue eye handle on its halo). From the text category, drag out the assignment part of 'Alphabet's cursor'. When you drop it, it becomes a script. From the script's menu, get a random tile and drop it in. Type 27 into it to choose one of the 27 positions in the Alphabet text.

```
Alphabet randomLetter ! normal
Alphabet's cursor ← random 27
```

Try it by pressing ! in the script, or this button.

See the random letter in this watcher.

Alphabet's characterAtCursor = b

Goal methinks it is like a weasel

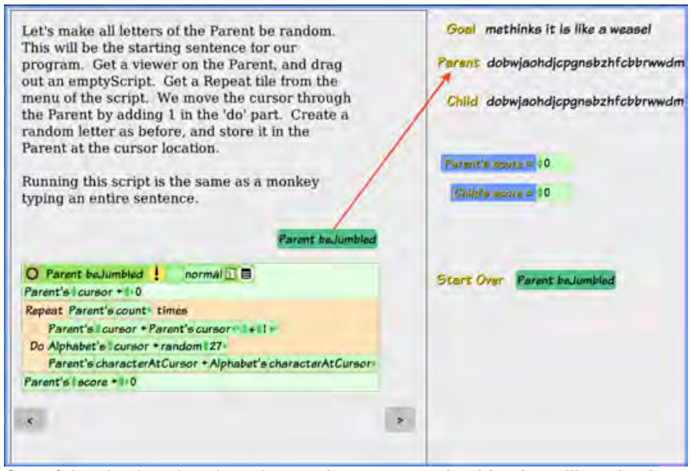
Parent dobwjaohdjcpgnsbzhfcbbrwwdm

Child dobwjaohdjcpgnsbzhfcbbrwwdm

Parent's score = 0

Child's score = 0

Instead of just "telling them what's so" or making claims, we can show them how to build their own simulation of a seemingly improbable task.



Let's make all letters of the Parent be random. This will be the starting sentence for our program. Get a viewer on the Parent, and drag out an emptyScript. Get a Repeat tile from the menu of the script. We move the cursor through the Parent by adding 1 in the 'do' part. Create a random letter as before, and store it in the Parent at the cursor location.

Running this script is the same as a monkey typing an entire sentence.

```
Parent beJumbled ! normal
Parent's cursor ← 0
Repeat Parent's count times
     Parent's cursor ← Parent's cursor + 1
  Do Alphabet's cursor ← random 27
     Parent's characterAtCursor ← Alphabet's characterAtCursor
Parent's score ← 0
```

Goal methinks it is like a weasel

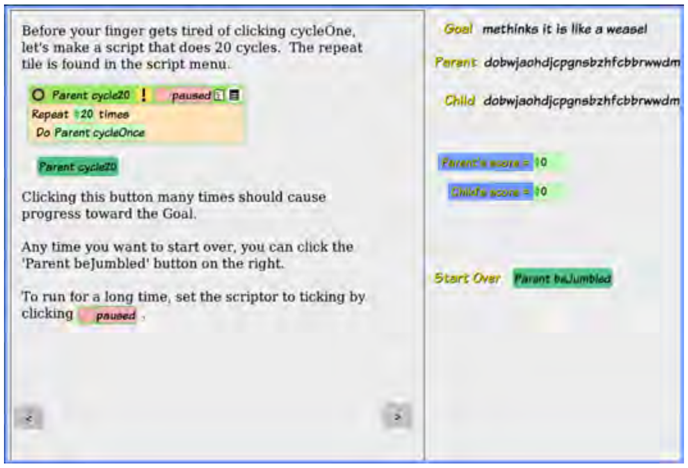
Parent dobwjaohdjcpgnsbzhfcbbrwwdm

Child dobwjaohdjcpgnsbzhfcbbrwwdm

Parent's score = 0

Child's score = 0

One of the simple subprojects is to make a sentence jumbler that will randomize the letters in a string of characters.



Before your finger gets tired of clicking cycleOne, let's make a script that does 20 cycles. The repeat tile is found in the script menu.

```
Parent cycle20 ! paused
Repeat 20 times
 Do Parent cycleOnce
```

Clicking this button many times should cause progress toward the Goal.

Any time you want to start over, you can click the 'Parent beJumbled' button on the right.

To run for a long time, set the scriptor to ticking by clicking paused.

Goal methinks it is like a weasel

Parent dobwjaohdjcpgnsbzhfcbbrwwdm

Child dobwjaohdjcpgnsbzhfcbbrwwdm

Parent's score = 0

Child's score = 0

Here's the payoff page. Quite surprisingly, the tiniest amount of retention of desired characteristics changes the problem from one that would take longer than the age of the unverse to do into one that our little Etoys simulation can do in a few thousand iterations (a minute or so on the OLPC)!

That’s it for this round. Much more to come this summer. We hope this helps you to get a feel for Etoys simulations and media – and, that you can see why the children like to make things in Etoys.