Etoys and Sim & Stories

Alan Kay, Kim Rose, Dan Ingalls, Ted Kaehler
John Maloney, Scott Wallace

VPRI Paper for Historical Context

Viewpoints Research Institute, 1025 Westwood Blvd 2nd flr, Los Angeles, CA 90024 t: (310) 208-0524

Etoys & SimStories

Alan Kay, Kim Rose, Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace
ImagiLearning Group, Corporate R&D, Walt Disney Imagineering

Summary

Etoys are computer environments that help people learn ideas by building them, and playing around with them. They help an enduser—usually a child—create a satisfying and fun computer model of the idea and give hints for how the user could expand the idea. SimStories are similar, but —like essays—string several ideas together to help the learner produce a longer and more concerted project.

This is the master design document for Etoys. As such, it will be constantly changing and will probably never be completely consistent. However, it should provide a useful gist of much of what we are planning to do for the rest of this year.

What's New In This Document as of February 26, 1997?

Everything! All is "in progress"; the last several sections are the most obviously unfinished.

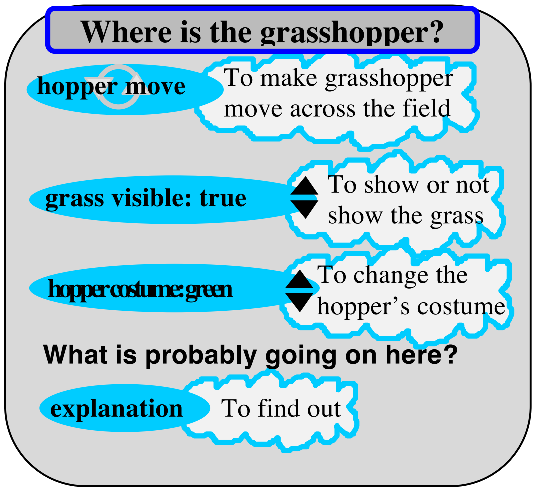
Contents

Etoys & SimStories

by Alan Kay, Kim Rose, Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace
ImagiLearning Group, Corporate R&D, Walt Disney Imagineering

Etoys[1] are computer environments that help people learn ideas by building them, and playing around with them. They help an enduser—usually a child—create a satisfying and fun computer model of the idea and give hints for how the user could expand the idea. SimStories are longer versions of Etoys that—like essays—string several ideas together to help the learner produce a deeper and more concerted project. A good motto for Etoys and SimStories is: *We make, not just to have, but to know*. Another motto that applies here is: *Doing with images makes symbols*. That is, the progression of learning moves from kinesthetic interactions with dynamic images to a symbolic expression of the idea.

Below is a simple Etoy that helps show why variation is important in Nature, and why a highly sensitive visual perception of movement evolved in most animals. The child has painted an outdoors scene with grass. Lurking in the grass is a grasshopper that seems completely invisible, even though it is really in plain view. A push on the movement button will reveal this immediately—when the grasshopper starts to move, we see it much better. A push of the “grass” toggle will make the grass visible or transparent, and provides another way of appreciating how camoflage works, and why Nature adopts it. Finally, we can change the grasshopper’s “costume” to see how much more visible it is when it is wearing a brown suit instead of green. A nice further phase of the project is to draw a brown dusty area in the scene and have both grasshoppers move over both the grass and the dust.


Where is the grasshopper?
hopper move
To make grasshopper move across the field
grass visible: true
To show or not show the grass
hopper costume: green
To change the hopper’s costume
What is probably going on here?
explanation
To find out

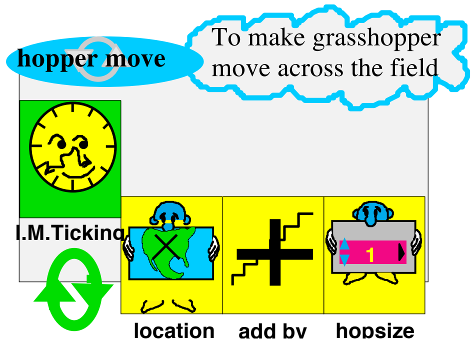
It is important to reemphasize that this (and all Etoys) are to be mostly constructed by the child users. The result, as we see, is like a small science project, in which an idea is modeled and then explored. In this example, the child is encouraged to draw the background, the grass, the grasshoppers, and to write the scripts for the four buttons that provide the animated content for

[1] We wish to thank David Vogler of DOL for his idea of “Dtoys”—interactive digital toys—that inspired our notion of “Etoys”—interactive digital toys with an educational twist that are constructed by the children.

the project. This particular project is within easy range of third graders, and many second graders.

One of the most important things about Etoys is that they start with kinesthetic and figurative (iconic and auditory) thinking, and then important parts of them are grounded in a clearly stated symbolic form. The scripts[2] have the same important relationship to science as classical mathematics has in the past; they are written in the language ***Squeak*** that was especially created for this project and will be discussed in more detail later in this note.

For this "Where is the grasshopper?" Etoy, a lot of the interest and knowledge is held iconically, but there are a few important symbolic parts. For example, let us "pop the hood" of the simple **hopper move** script that the child wrote to animate the grasshopper:



We see that a script is the action giving part of a button. There are two sections to this script. The first—listen to the clock ticks—causes the script to be executed whenever the main clock ticks. (This clock, which is called **I.M.Ticking**, is the one that controls all of the animation in the child's world.) The second part of the script is a request that asks Squeak to change the **location** property (of the grasshopper costume in the scene) to 1 plus its old value. This will be done on each tick. The result will be that the grasshopper costume will animate smoothly to the right. The scene automatically wraps around unless told differently, and thus the grasshopper will move off the scene to the right and reappear on the left. This is sufficient for this Etoy.

This script seems simple and harmless enough, and it is. But, it is also the gateway to some really amazing and important ideas to be encountered later on, including how gravity works, synthesizing music, and graphical effects.

Since young (9 years old) to very young (3-4 years old) children will be writing scripts for their characters, the scripts are fashioned from a collection of tiles, each of which has both a picture and a descriptive word. These tiles are "pulled" in various ways from the user interface with only very occasional typing required.

There is another, very important, part of each Etoy that we haven't shown yet: the curricular scaffolding that helps the child build and understand an Etoy. This scaffolding merits an in-depth discussion which has been placed later in this note.

[2] The scripting language used in this note is an approximation to the one that will be used in the released Etoys. Squeak has the ability to change language forms easily and this leads to experimentation.

The Place Of Etoys Within Human Knowledge And Learning

Etoys are "science and math" in the large. By this, we mean that science is much more than "scientific knowledge"—it is not only a collection of practices for being able to find out things, *but a way to gain some sense of how accurate the findings are likely to be*. Mathematics is one of science's most important practices. Here I take the larger meaning of mathematics as *the art and skill of being able to reason*—that is, to be able to take representations in a language and to show how they are related to other representations in a language. One can see how important this is, since all we can have inside our heads are ideas represented in various languages. None of these have anything necessarily to do with the real world. They are just stories—or to look at them a more neutral way: maps. Stories can be perfectly logical and still have nothing to do with the world. A map can be drawn of a place that has never existed. We can reason from them. They can be logical or not, and so forth.

What science has tried to do is to use our most dangerous facility—reasoning—in its most dangerous form—mathematics—in a way in which most of the strengths of reasoning are realized and most of its faults are minimized. The way this works is that no wonderfully reasoned conclusion is held to be "reasonable" until its consequences can be tested in many ways. Even then, it is realized that the conclusion cannot be "true" in the old time sense of the word.

This sounds ridiculously stringent and careful: too constrictive by far to yield anything. Yet science has been endlessly more successful in "seeing" the world than any other method. And it is less than 400 years old. There are many belief systems on earth: not just for every culture, but for every human. But for a 12 year old who has just contracted diabetes, there is only one that "knows" what is wrong, and "knows" how to make the molecules that are needed. This in spite that this belief system—science—does not really "know" what a molecule (or electron, etc.) is.

This also sounds somewhat unromantic. Where are the dragons, the sorcerors, the swordsmen and women, the magical spells and potions? Yet, science is the most romantic pursuit of the 20th century and most scientists and enthusiasts feel this romance as keenly as any New Ager. It retains the essential mystery that Romance requires, but tries to do away with mysticism. Even better, it works so well—not just for the 12 year old with diabetes, but in so many other areas of life we hold most important.

A central point is this: In the Middle Ages reading and writing was mostly for professional scholars. Now it is held to be a basic human right and pursuit. We don't expect everyone to become a scholar via reading—we do hope that they will expand their minds to be able to touch much more than their commonsense world deals with. Likewise, learning scientific and mathematical ways of thinking are not just for professionals, but for widening the horizons of all, for aiding the thinking of all, for helping all of us to wake up a little more. One concrete example is that constitutional governments were *inventions* brought forth in the seventeenth and especially the eighteenth centuries as careful attempts to apply scientific-style reasoning to the problems of civics and governance. They require *science*—both careful observation and careful reasoning—to work.

We have just expressed one of the things that we think children *need* to learn. Children, generally, *want* very different things: they want to play, to be social, to fantasize, to listen to stories, to explore and control their world, and so forth. Yes, they want explanations about how the world works, but over the centuries they have been generally quite content to accept almost any collection of stories as fact. Maria Montessori was the genius who saw how to let children exercise their *wants* and still get them to learn what we think they *need*.

Her great idea starts with the observation that children are driven to learn their immediate environment and culture through play—much of it via physical artifacts. She realized that what children *want* is not always the same as what they *need*—especially in 20th century Western culture. For example, children have built in drives to learn to speak, but not to read and write; there have been many cultures in history that are illiterate. Writing and the reading of it was an *invention*, not something built into people to do. Her genius was to devise a special environment and artifacts that look like toys to children—catering to their *wants*—but with beautiful side effects that help them learn what we adults think they *need*. This was and still is a great idea.

Another of her insights, which was taken up in more detail by Piaget (a great admirer of hers), was that young children need lots of direct tactile contact with many of the things they are trying to learn. Jerome Bruner once noted that a generally good approach to learning human inventions and discoveries at any age is to start off with kinesthetic experiences that connect with iconic and other figurative experiences (such as organized sounds: music, etc.), and find some way of culminating in symbolic form—that is, in written language, mathematics, notated music, etc. Seymour Papert realized that the computer was a brand new world in which "math" was the native language—and that there were fruitful prospects for teaching children to think better by getting them to embody their ideas as programs which could be run and tested. Our own work in the past has shown that the object-oriented simulation style is a very strong way for children to deal with "powerful ideas".

Etoys are a direct reflection of these ideas.

Putting Etoys In Context

Several thousand Etoys would start to look a lot like active narratives and illustrations in an electronic encyclopedia. We can imagine that most articles in such an encyclopedia would be a combination of narrative, illustrations, and symbolic descriptions of models to be completed by the reader. For example, the simple camoflage Etoy described above would be one of several that would be found in an article on variation of species (and within species) in Nature. This is one of our aims over the next several years.

Most of the Etoys are non-gender-specific, though no special care has been taken to ensure this. If this becomes a larger issue, then more work can be done in the scaffolding to provide more options.

For now, we plan to release Etoys sequentially, as they are made, for each of the Daily Blast, Premium, and Family.com services. Each of these suggest different contexts for presentation. For example, an "Etoy A Week" could be a feature on the Daily Blast—after its week, the Etoy could then migrate to the Premium service to become part of the Kid's encyclopedia. Etoys without special artwork and other production values could be routinely released to Family.com and then later be repurposed with higher production values for the paid services.

An exciting Etoy would be a Dtoy that is deconstructable. That is, the Dtoy is already made up to do its thing, but children can also "pop the hatch" to see how it works and make variations of their own. There are several current Dtoys that would be good candidates for deconstruction, for example: "Math Class Mayhem" (which shoots spitballs), "Dragable Dan" (whose limbs are elastic and can be dragged), "Flying Letters" (in which submitted sentences will follow a path draw by the mouse to produce a "modest trail and becomes a kind of magical 'shooting star'"[3]).

[3] **D-TOYS CREATIVE SUMMARY**, David Vogler, 3 February 1997

The First Dozen (or so) Etoys

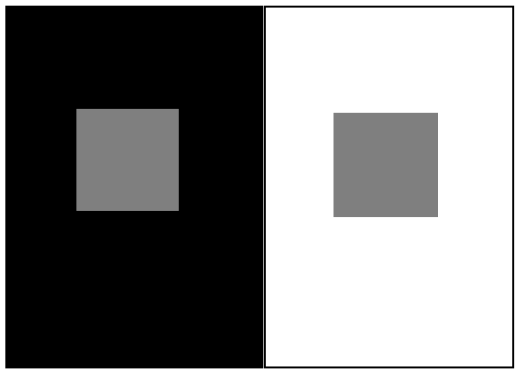
These are given in no special order (except that Etoys that supply ideas needed in later Etoys are given earlier). The illustrations are simply schematics for now, and are only intended to give a sense of what the Etoy will actually look like.

1 • Where Is The Grasshopper?

What The Children Do
The children get to draw and paint the environment, grass, and the grasshoppers. They will also write simple scripts that will:
- move the grasshoppers horizontally in and out of the grass
- make the grass visible or not
- change the grasshopper's costume from green to brown

What Is Learned
- why nature uses camoflage
- why motion sensing is very important
- how to move an object with uniform velocity

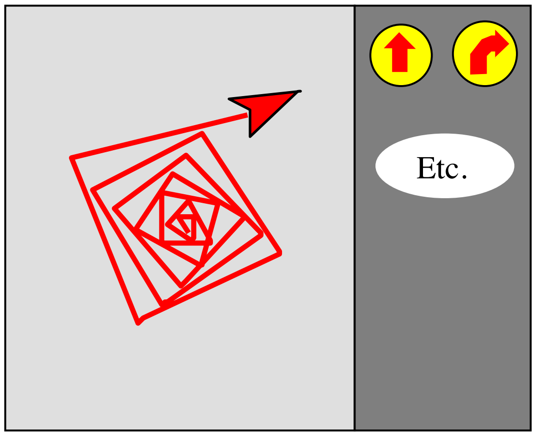
2 • Which Is Darker?



What The Children Do
The children draw and/or paint the environment and squares. They will also write simple scripts that will:
- move the squares around the canvas at different speeds to see how they interact with each other and if the instant they appear to change color can be determined
- change the colors of the background and squares to see what those effects contribute

What Is Learned
- things are not as they seem
- our vision system enhances contrast
- how to move an object with uniform velocity

3 • Turtle Button Box



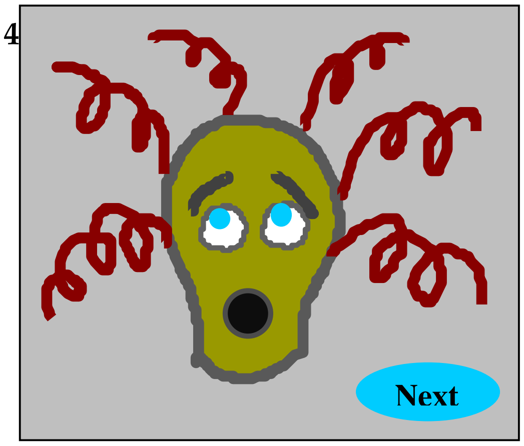
What The Children Do
This is especially nice for younger children.

- They control the turtle(s) through the "button box" to move forward, turn, change color, pen up or down, and most importantly, to remember their sequence of actions.
- The action sequences (scripts) can be edited and replayed. More complicated designs can be made with the action sequences (like the "Squiral" to the left).

What Is Learned

- how to imagine at a distance
- numbers are really useful for doing graphics and iterations
- physical actions can be written down and edited
- parameters, iteration, etc.

4



What The Children Do

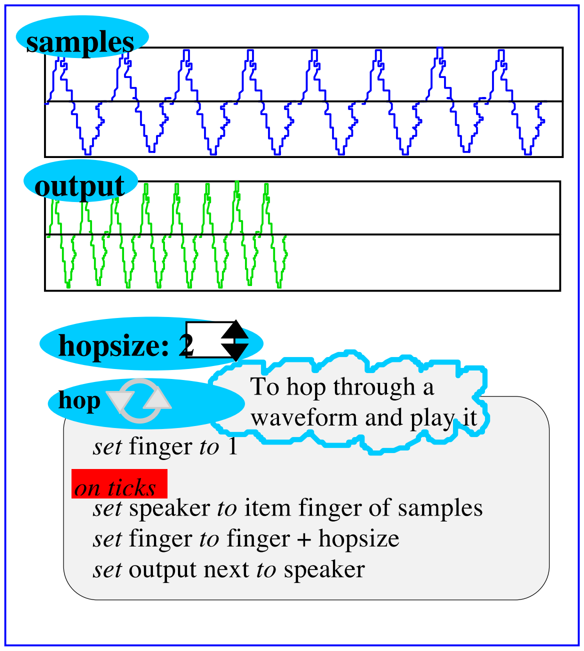
- Draw all the hair-doos and facial expressions. (They are given a notebook of ideas, but it is only for reference, it can't be copied from.) These are all put in a Squeak object/folder.
- write the script for the **next** button that first just moves through the doos in order, and then selects from them randomly.

What Is Learned

- how to draw different facial expressions
- how to sequence through a list of items
- how to randomly select from a list of items
- parameters, iteration, etc.
- planning, etc.

[4] This is adapted from a Dtoy

5 • Make Your Own Music



```
set finger to 1
on ticks
  set speaker to item finger of samples
  set finger to finger + hopsize
  set output next to speaker
```

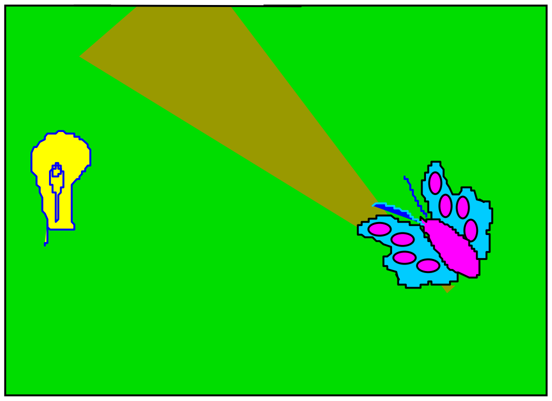
What The Children Do

- Either record sound samples through microphone (this is easy on a Mac, and can be difficult on a PC) or use supplied sound samples (of "things around the house", people, hit things, etc.
- write several scripts that **hop** through the samples and send them to the speaker. The first simply goes through in order (with a **hopsize** of **1**), and looks just like the script for animating in one dimension. The second script makes the **hopsize 2** and the result is the same sound an octave higher. All pitches can be synthesized with the appropriate **hopsize**! The third script (shown) also puts what goes out to the speaker in a container so we can see that its length is always **1/hopsize**.

What Is Learned

- how synthesizers work—they are just like animation except though a wave table—this is another example of the "powerful idea" of *differential models*
- how to sequence through a list of items
- parameters, iteration, etc.

6 • Find The Light



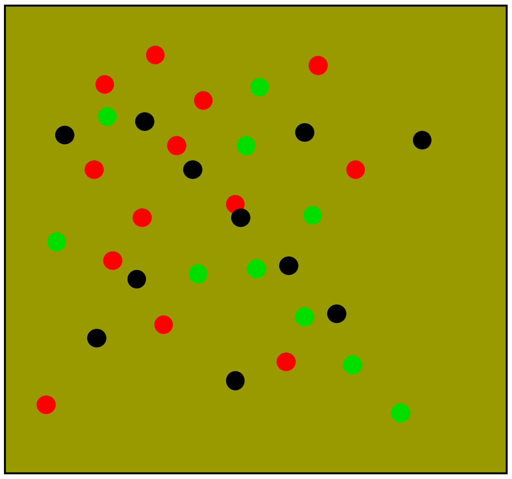
What The Children Do

- Draw several lights, a creature (such as a butterfly, turtle, bacterium, etc.), and a "sensor", a triangular form that will provide the creature's "cone of vision". This is hooked to the creature, will be invisible most of the time, but can be made visible for debugging.
- write a script that will return **true** if the sensor collides with a light, and **false** otherwise.
- write a script that will make the creature go **forward** if it can "see" the light, and will **turn** otherwise.
- The above will always find a light within range of the sensor. The children can experiment with several lights, several creatures, and best of all, creatures with lights on them. Putting the creatures' pens down with make really interesting tracks.

What Is Learned

- How the *powerful idea*of *feedback* works.
- how to put sensors on objects

7 • Are You Infectious?



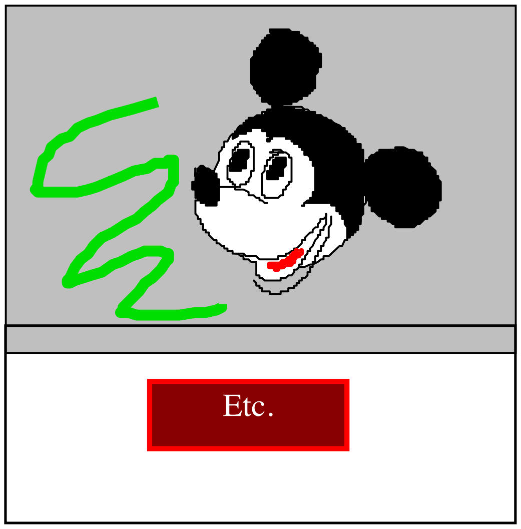
What The Children Do

- Make hundreds of objects that all have the same behavior: they move randomly around the canvas, when they collide with another they check to see if either one is infected; if only one isn't then the other becomes infected; if one has been infected for some number of ticks it dies, turns black. stops, but can still infect.
- Later, they hook up a graph to the state of the objects to see just how fast an epidemic can take hold.

What Is Learned

- How the *powerful idea*of *exponential* growth works.
- how to use collision detection.

8 • Make Your Own PaintSet



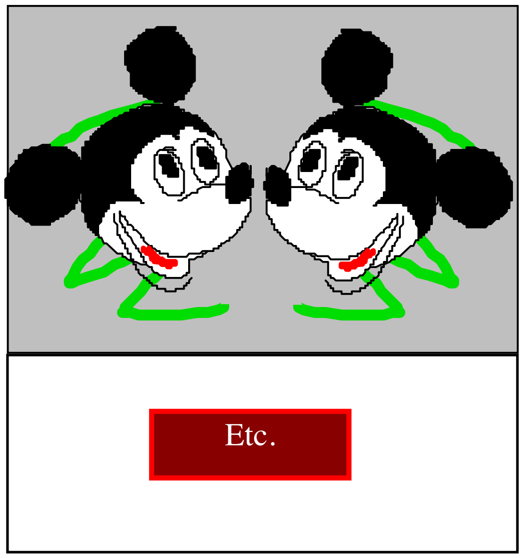
What The Children Do

- All objects have pens, so a simple painting system is just a little bit more work than simply taking the mouse's location, moving an object to it, and stamping the object's costume into either the background, or into the "painting capture" layer. This is a very satisfying project for most kids; it not only gives them a sense of mastery over a basic tool, but also removes some of the mystery that lies under their interface.
- Later, they can add more colors, and then start to program effects (see "Kaleidoscope" ahead).

What Is Learned

- How the mouse can be used to move and point to things
- How painting works

9 • Mirror Painting



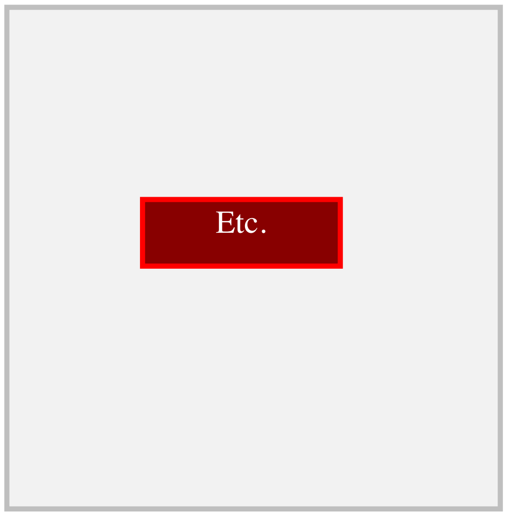
What The Children Do

- This starts with the kids' painting system (we use the previous project as a prerequisite). All we have to do to get mirror painting is to set up an additional pen and give it the same mouse locations, but with the *leftright* coordinate negated (times -1). This is very satisfying and powerful.
- Later, they can organize four pens, each in a different quadrent, and finally, many different pens in lots of different arrangements.

What Is Learned

- What minus actually does
- The quadrents

10 • How Does Color Work?



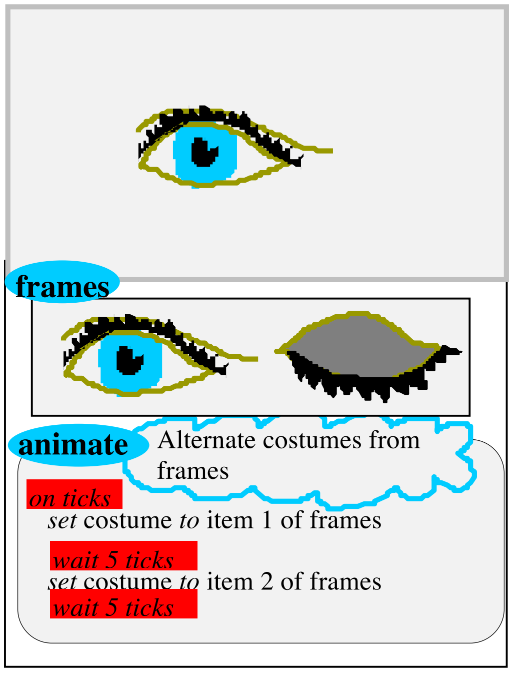
What The Children Do

- First, in the real world, we have the kids do some experiments with colored transparencies, lights, and paints. (This is because color, and the way we perceive it, are physical and psychological phenomena that are properties of the universe and not predictable.)
- Then, they write scripts that will mix colors for them—as the real world does. This can be part of their painting program or as a separate project.

What Is Learned

- How we perceive color as a mixture of primaries.
- How to mix colors so that our eyes will perceive them correctly.

11 • Two Frame Animations



```
on ticks
  set costume to item 1 of frames
  wait 5 ticks
  set costume to item 2 of frames
  wait 5 ticks
```

What The Children Do

- These are the essence of all animation—the control of "what happens *between* the frames". A good one to start with is a blinking eye. The two frames are easy to draw since the Squeak painting system greys out what is already on the screen—so the open eye can be used to register the closed one.
- The strategy here is similar to that used in the HairDoo Etoy: use a folder to hold the alternatives, and then write a script to choose the costumes in turn. The *wait 5 ticks* is needed to slow down the frame rate to that of a reasonably blinking eye.

What Is Learned

- The "phi phenomenon" of human vision (which allows us to experience animation as continuous movement) works really well even at very low frame rates.
- The start of a strategy for doing more elaborate animations.

12 • Make An Animated Flying Bird

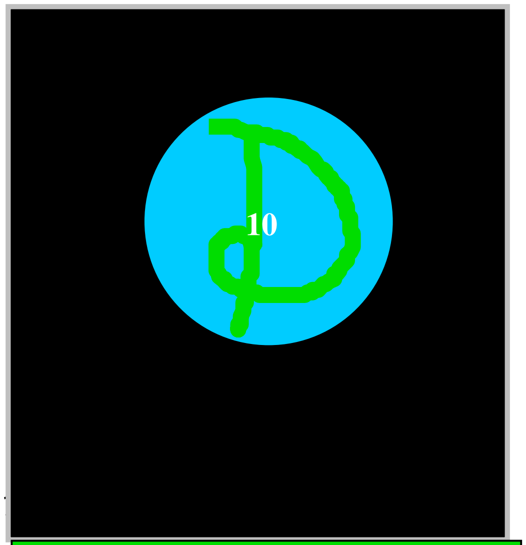
What The Children Do

- We supply a "notebook" that shows a bunch of four frame animations to help the child draw the frames.
- The frames are put into a frame folder as in the two-frame animations.
- The script could be adapted from the two-frame Etoy (and perhaps should be for a first shot).
- The next script should be one that will take care of arbitrary numbers of frame cycles.
- Finally, the child can experiment with a folder of frame folders, to vary the movements.

What Is Learned

- How animation cycles are done in "pro" systems
- How to make a real animation tool

13 • Reverse Your Color Vision[5]

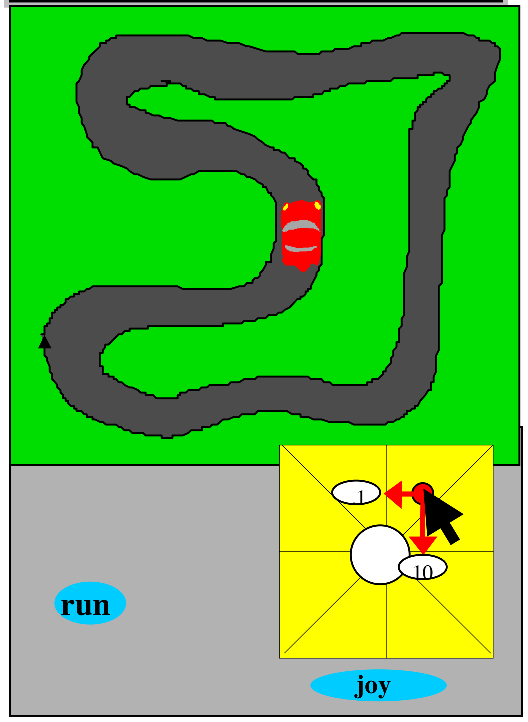


What The Children Do

- A good one to start with is the DOL "D-Ball". The child draws it in blue with green lettering.
- Writes a script that does a countdown of ten seconds and then makes the background white and the D-Ball invisible. The result is to see the D-Ball's reversed after image in its normal colors.
- Tries other paintings, etc.

What Is Learned

- Another interesting facet of persistance of vision and filtering overshoot.
- A few more useful techniques in how to control events over time.



What The Children Do

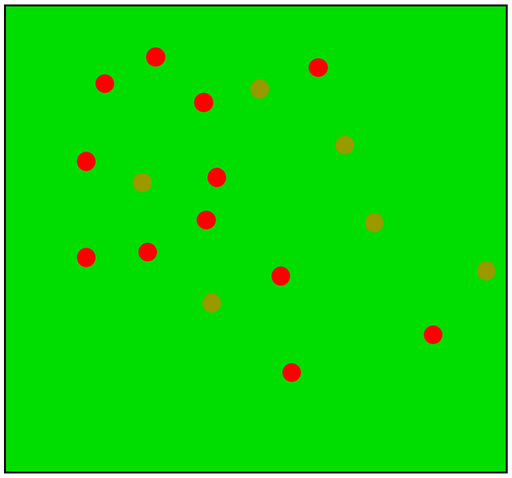
- This can use a real joystick or a simulated one (that is supplied). All objects act like LOGO turtles, so, to start, the children simply draw the vehicles of their choice (a notebook of suggestions is supplied). They can then "drive them" via typing scripting commands like *forward 1* and*right 30*, etc.
- The first script will move the car forward and back by sending the joystick's *updown* outputs to *forward*.
- The second will add turning to the car by sending the joystick's *leftright* outputs to add to *heading*.
- One game can be to keep the car on the road without the collision alarm sounding; another might be to add another car and joystick for another player.

What Is Learned

- This is another extension of "turtle graphics" (what adults call "differential geometry in intertial coordinates).
- This helps to displace one's own point of view and to see how the world looks from another dynamic perspective.

[5] Adapted from a Dtoy

15 • Pandemonium



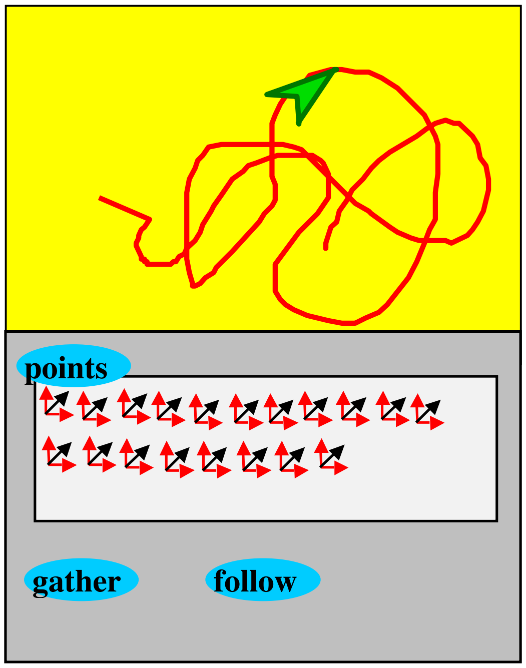
What The Children Do

- This is a “bouncing off things game”. The first project is to set up walls and get the objects to bounce off them. The child is encouraged to try rolling a wet ball against a wall and to measure the angle coming in and going out. They are the same.
- This observation has to be translated into turtle terms. This problem is really helped by using the *leftright* and *updown* components of the object’s velocity. Then the child will see that running into a vertical wall needs the *leftright* part of the velocity to be negated, and for a horozontal wall, it is the *updown* component that has to be negated.
- This is lots of fun with zillions of objects.
- A further wrinkle is to control the ability of each object to bounce, by scaling the velocity.

What Is Learned

- That the “angle of incidence is equal to the angle of reflection” is sometimes called Snell’s Law.

16 • Fly An Object In A Path



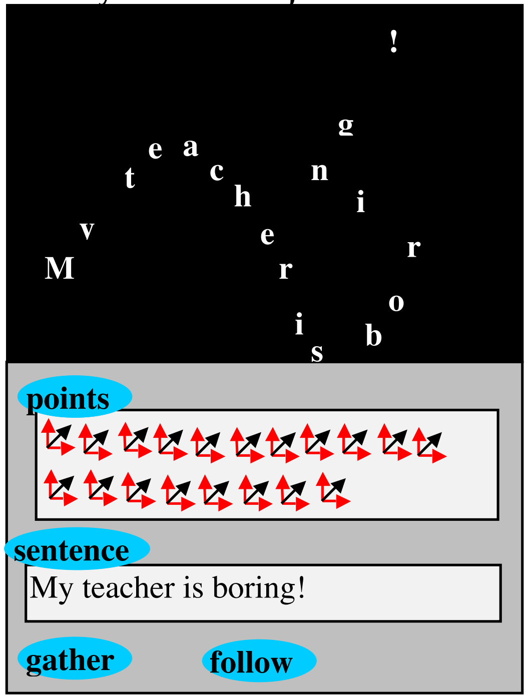
What The Children Do

- The children write a script that captures the mouse coordinates into a folder.
- Then they write a script that animates any object over the path. This is particularly rewarding when applied to the animated bird flapping its wings.

What Is Learned

- How to use collections of objects as “streams in time” to control other objects.

17 • Fly Lots Of Objects In A Path[6]



My teacher is boring!

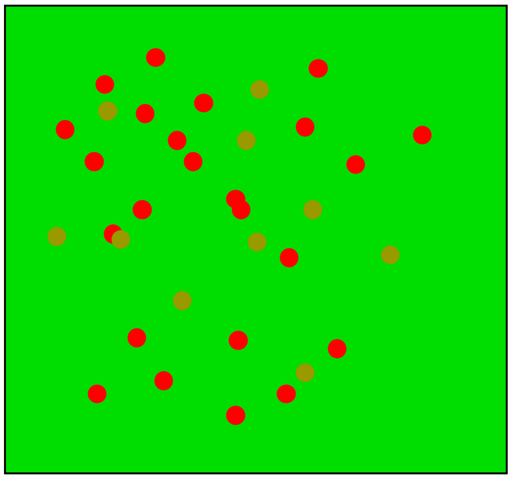
What The Children Do

- This works like the previous “follow the path” Etoy. What is new is to type a sentence into a folder and to use it as a source of objects.

What Is Learned

- That everything in Squeak is an object, even little text characters.
- That text is actually a “collection folder” like all other collection folders.
- How to organize what seems to be a complicated project into just a few lines of script.

18 • Foxes and Rabbits



What The Children Do

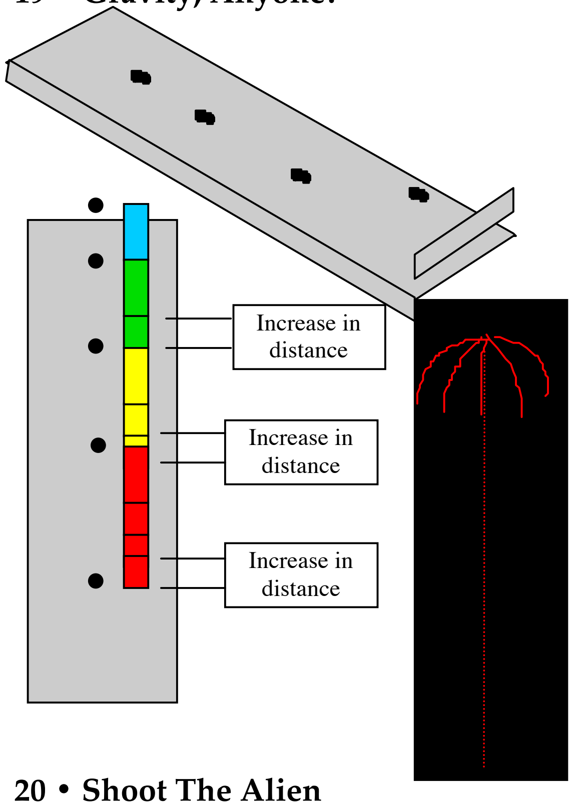
- The child starts off by making a few foxes and lots of rabbits. These are all small colored objects.
- Writes a script that will decease a fox if it goes for so many ticks without food.
- Writes a script that will decease a rabbit if it contacts a fox or if it gets old enough.
- Writes a script that will make more foxes if they have eaten enough.
- Writes a script that will make more rabbits periodically.

What Is Learned

- Sometimes they all die, but usually the rises and falls of the populations alternate as the foxes overeat and kill off their food supply, thus perishing themselves, which allows the rabbits to recover, etc.

[6] This is adapted from a Dtoy

19 • Gravity, Anyone?



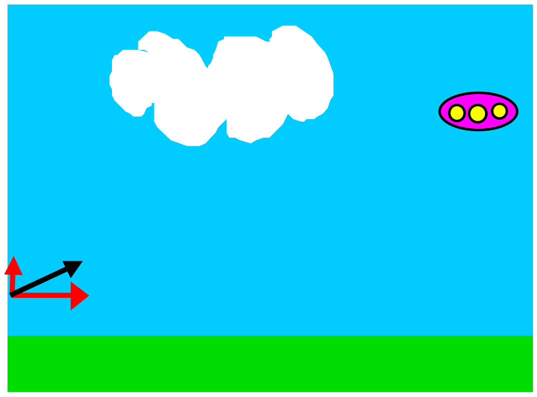
What The Children Do

- This Etoy is large enough to be a SimStory, but focuses on just one idea: that gravity “eats velocity”. Since gravity is in the real world and works the way the real world works, we need to do an experiment to see how it behaves.
- We encourage the child to actually do this experiment—but, just in case, we have a simulation of it. A toy truck with a baggie full of water with a hole pricked in it is sent down a ramp. The drops hit in equal time intervals but gravity somehow makes the truck go faster and faster, so the drops are not equally spaced.
- We measure the drops and disover that the added distance between each drop to the next is the same. This leads to a simple script that will accellerate the toy truck.
- A further wrinkle is to draw rockets and shoot them upwards with a high initial velocity. Gravity is accellerating in the opposity direction, gradually “eating” the velocity until it turns negative and starts to fall.

What Is Learned

- The above: what adults call the “2nd order differential equation” for gravity.

20 • Shoot The Alien



What The Children Do

- This builds on the previous Etoy. The only addition is to establish the idea that the velocity will stay constant in a direction if there are no forces acting on the object in that direction.
- Again there is an experiment, which is to find the smoothest surface, start something moving, and convince ourselves that the only reason we see slowdown is because of friction.
- If we think about how a cannon works: it starts the shell with a high initial velocity. Gravity will eat at the *updown* component, and “nothing” will eat at the *leftright* component (we are ignoring air resistance). This leads to a simple script for the cannon.
- We find what angle will shoot the longest. Then we try dropping an alien and see if we can hit it. The answer is surprising.

What Is Learned

- The above, plus something special about gravity.

"The Typing Monkees": A SimStory About How Evolution Works

This "SimStory" or "Active Essay" has been constructed by us several times over the last 2 years. We present it here using the same style as the previous Etoys.

<TO APPEAR IN NEXT RELEASE OF THIS DOCUMENT>

“Themes Of The Week” And Related Etoys

The Premium service could have a “Theme Of The Week” in which a specially constructed Etoy could play a part. Following are some examples and suggestions for this form of presentation:

And the feature of the week is...

- **Space**

The space between your teeth, the space we mean when we say, “I need my space” and outer space - that place Dr. Spock and Capt. Kirk hang out, living in space, sattelites, pictures of space, the planets, Holst, space missions, astronauts

related Etoy -- orbits and planets

- **Blue**

From Picasso’s Blue period to why is the sky blue, the blue lagoon, once in a blue moon, blue bayou, singin’ the blues
Can be done with any single color -- The Color Purple - Harold’s Purple Crayon, I never saw a purple cow, where does purple come from? Why do we call it purple? are grapes puple?

related Etoy - how colors mix- paint color/light color

- **The Four Seasons**

Vivaldi, gardening, weather, the northern and southern hemispheres, what season is it for you? Poems and essays

related Etoy - what makes the seaons change?

- **Water**

The ocean, lakes, saltwater, freshwater, the water in our bodies, H2O, how do fish breathe in water? pond ecology, a drop of water, how many drops of water fit on a penny? raindrops, tears, waterfalls, water flow, river rafting, how much water is in a swimming pool?, aquaria, how does water get to our kitchen faucets and showers? where does the water go from the drain?

related Etoy - build a drainage system, a wave-making machine, what happens when water boils and freezes?

Note: Possible Bill Nye tie in with various Quicktime movies in his “video lab/Media lab”. (Both times I tried to download a Quicktime movie to watch, my Mac bombed.)

- **Brains**

thinking, animal brains, human brains, brainy people, what is the moosh our brains are made of? what is information? knowledge? data? thinking machines, artificial intelligence

related Etoy -- Eliza?? can machines think? or a stimulus response machine

- **Money**

currency exchange, how coins are minted, collateral, trading, budgets, money from other countries, the stock market, interest, interest rates, how do credit cards work?

related Etoy - interest & compounded interest

- **Time**

what time is it where you are? how do time zones work? who made the first clock? what did people do before we had clocks, how do sundials work?, calendars for timekeeping, Time Travel - is it possible? Age? What makes us age? Why do we never have enough time? what if we could stop time? telling time? Why do we have 24 hours in a day? the sands of time......Einstein

related Etoy - make a clock from logic (doorbell) type

- **Toys & Games**

in history, what was your mom’s favorite toy? Your grandfather? What did Roman kids play with? How about kids in the Renaissance? games through history/the ages/from other countries, who made the first stuffed animal? Talking toys....Chatty Cathy to Woody and Buzz Lightyear

related Etoy - make a 'pong ' or 'breakthrough' game

• **Architecture & Buildings**

how do buildings stand up? who designs buildings? exteriors? interiors? what is an architect? famous architects, famous buildings from your city, what is old? modern? skyscrapers, stories about buildings, design your own house or building, bridges, factories, office buildings -- what really goes on?, styles, movements, trends, how do elevators work? how does music come into an elevator? why is it always so bad? who invented the cubicle? who was Herman Miller?

related Etoy - how high is your school? your house?

• **Things that Move**

........from molecules to rocket ships

on the go, always moving, planet earth to your fingers when you play the piano or click a mouse, what keeps us going? how to earthworms move? how do airplanes stay in the air? moving pictures, trains, escalators,

NOTE: BILL NYE TIE IN - "DEMO OF THE DAY":

Momentum In Action

When objects are moving, they have momentum. Their momentum depends on how much things weigh and how fast they are moving. Want to see momentum in action?

Then try this experiment:

MATERIALS:

1. A ruler.
2. A quarter.
3. A dime.

DIRECTIONS:

1. Put the ruler on a table.
2. Put the dime at one end on the ruler. Make sure the dime is right up against the ruler.
3. Flick the quarter at the other end of the ruler and watch what happens to the dime.
4. Now, put the quarter at one end of the ruler.
5. Watch the quarter as you flick the dime at the other end of the ruler.

Moving things have momentum, even the flicked quarter, and momentum can be transferred from one thing to another. The flicked quarter transferred momentum through the ruler and to the dime. The flicked dime transferred momentum too, but lighter things have less momentum than heavier things. That's why the quarter didn't go quite as far as the dime.

related Etoy - motion via animation scripts or via experiements, i.e. a truck rolling down a slope, acceleration, velocity, speed

• **You've gotta have heart**

from aorta to amour....the human heart, how blood flows, veins, arteries, animal hearts, why do we think of love when we think of hearts, emotion, emotional intelligence, the queen of hearts, having a "crush", hearbeat, "in a heartbeat", the card game of hearts

related Etoy - build a human heart - how the heart pumps blood

• **Tubes, circuits, batteries & bulbs**

all things electric, battery operated, electronic, how do radios work? tvs? what's a vacuum tube? a transistor? circuits? cirucuit boards? what is silicon? what's on a chip? what makes a lightbulb work?

related Etoy - lots of possibilities

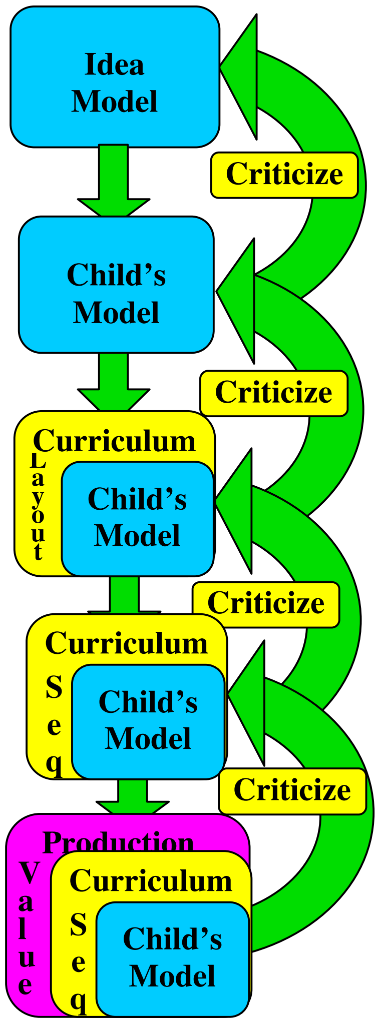
- **‘round and ‘round**

planet earth, oranges, bouncing balls, spheres, things that go ‘round and ‘round

Producing Etoys

There are two central difficulties in producing E-Toy projects. The first is to have the scripting demands be "just right" for the end-user. Etoys are educational, and this means in part that they should not be given to the child in the form of too easy prefabbed kits. Education is not a state of being but a manner of traveling. There are several important reasons—explained in detail further on—for having the child do serious scripting of objects. In other words, we want to have a *controllable* level of difficulty in the child's scripting. On the other hand, we don't want the child to get bogged down in the irrelevant details that often arise when programming in an unsuitable language. This first difficulty is mitigated by having the end-user authoring system that is used by the child be completely rendered in an underlying "authoring system authoring system" (ours is called Squeak). This allows quick additions and changes to be made to the child's tools as new needs arise so that the scripts the child writes have maximum power and minimum gratuitous difficulties.

The second, and major, difficulty, in making up an E-Toy project is the creation of a curricular scaffolding around the tools that will help the child to complete the project, but with the child taking as much of the lead as possible. Here is the progression of the production of an Etoy:



1. Get the idea going "any which way". This will often employ coding techniques that will be beyond the scope of the child. The main goal here is to find an engaging end result that really brings home the central thrust of the idea.

2. Render the Etoy model into the form that we hope the child will be able to build. This may require modifying the children's authoring system (by going deeper into the Squeak "authoring-system-authoring-system" to make necessary changes, add syntax, tools, etc.).

3. Do a "curriculum layout" of all the sequences and advice that are required by the end product. This includes linking to related items within the Internet that are of interest. This stage will include user testing with a human acting the part of the curriculum. The child's model may be found wanting and may have to be changed.

4. Create and install the actual "curriculum sequels" into the Etoy. The Etoy is now functionally complete and can be tested via the Internet. This stage should have high enough production values for Family.com, and could be released there.

5. The final phase is to add any additional production values that the release venue requires. For example, the Premium service may very likely require more finished art, Disney characters, etc. The Etoy is now ready for release.

Curricular Scaffolding

Now let us take a look at the process that children go though in making their Etoys, and ask ourselves how we could help a youngster to (a) stay interested in the project, (b) find suitable tools in the authoring system, and how we would (c) monitor progress in the project, (d) deal with questions that might arise, and (e) help with errors that are likely to be made.

A discussion of the actual user interface of this system is beyond the scope of this note. Instead I will try to sketch what has to be done, and roughly what tools there are to help the children accomplish their goals.

In the drawing part of the project, the child can create the visual elements in any order. All are done with a comprehensive graphics/painting/drawing interface than is an integral part of the child's authoring system. Hints are useful. For example, it is reasonable to hint to the child that grass is easily made by drawing a few strands and then copying and pasting them all over the field. The background can be simple neutral colors suggesting ground and sky. A cloud is easy to make and place. Some children will want to spend a lot of time on the visual part of this project, and some much less. This is to be expected. It *is* really important for the system to have a very high quality art interface, both for those who already want it, *and* for those children that might get a little interested in drawing and can be helped to get better at it. Thus, there will be hyperpointers off to other curricula for learning to draw and paint. It will be important for Disney to offer a good cheap tablet at a discount as part of its paid services, etc.

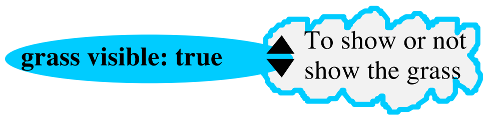
It is known that tracing is not very good for learning to draw, but that *copying*—that is, to use someone else's drawing as a guide—really does help. One trick that really works well for helping unskilled draftsfolk to learn to draw is to have them copy an upside down drawing. Thus, one of the things that should be in the curriculum tools is the ability to show the child a suggested drawing in a way that they can't trace it, and to give them the option to copy it upside down. Later on, I will say more about why learning to draw can be an important step in learning modern skilled thinking.

Now every time the child has drawn a figure, the system has automatically created an underlying *object* with the drawing as its current **costume**. The child has some sense of this because they can drag their drawings around—they are not embedded in the background. If the child double clicks (or the equivalent) on an object, it will reveal how it is made. Every object is a collection of *properties*, some opf which are simple: such as **location** on the canvas, or its **width** and **height** of the current **costume**; or more complex, such as a strategy for **moving** about on the canvas, what to do when something is **collided** with, and so forth. There are a lot of potential properties for each object, so the system intially shows just a few of the more used ones, with the rest kept in "drawers", each of which has a category name, such as: *Visual Appearence, Carrying, Moving*, etc.

Another part of the user interface is a *Toybox*. This contains some already made up useful objects, such as: editable text objects, numbers, sliders, buttons, etc. As many *instances* as needed can be dragged out on the canvas. All objects—including those that the child makes—have the same rules. Again the drawer metaphor will help to organize and find the existing object types. The children will use their *toyboxes* to hold the Etoys they make as well.

The two simplest buttons for our camoflage Etoy are the **grass visible** and **explanation** buttons. First we select all of the grass we have made and then *group* it into one object. We double click on this to get its properties. We find the **visible** property which is set to **true**. We use the little arrow to see what other values it can have, and choose **false**. The grass disappears. We choose **true** and the grass reappears. This will do the job if we can get it to our command area. We do this by dragging it (this makes a special kind of copy automatically) to our control area. When we let it go, its name automatically changes to the name we have given to the grass object plus the name of the property: **grass visible: true**. This is done so that we can keep track of who the button belongs to when one has been dragged out of its properties. The last thing we do is to close the *method* part of the button—this is the part that actually does the work—but we leave the button and its comment showing.

The result is:

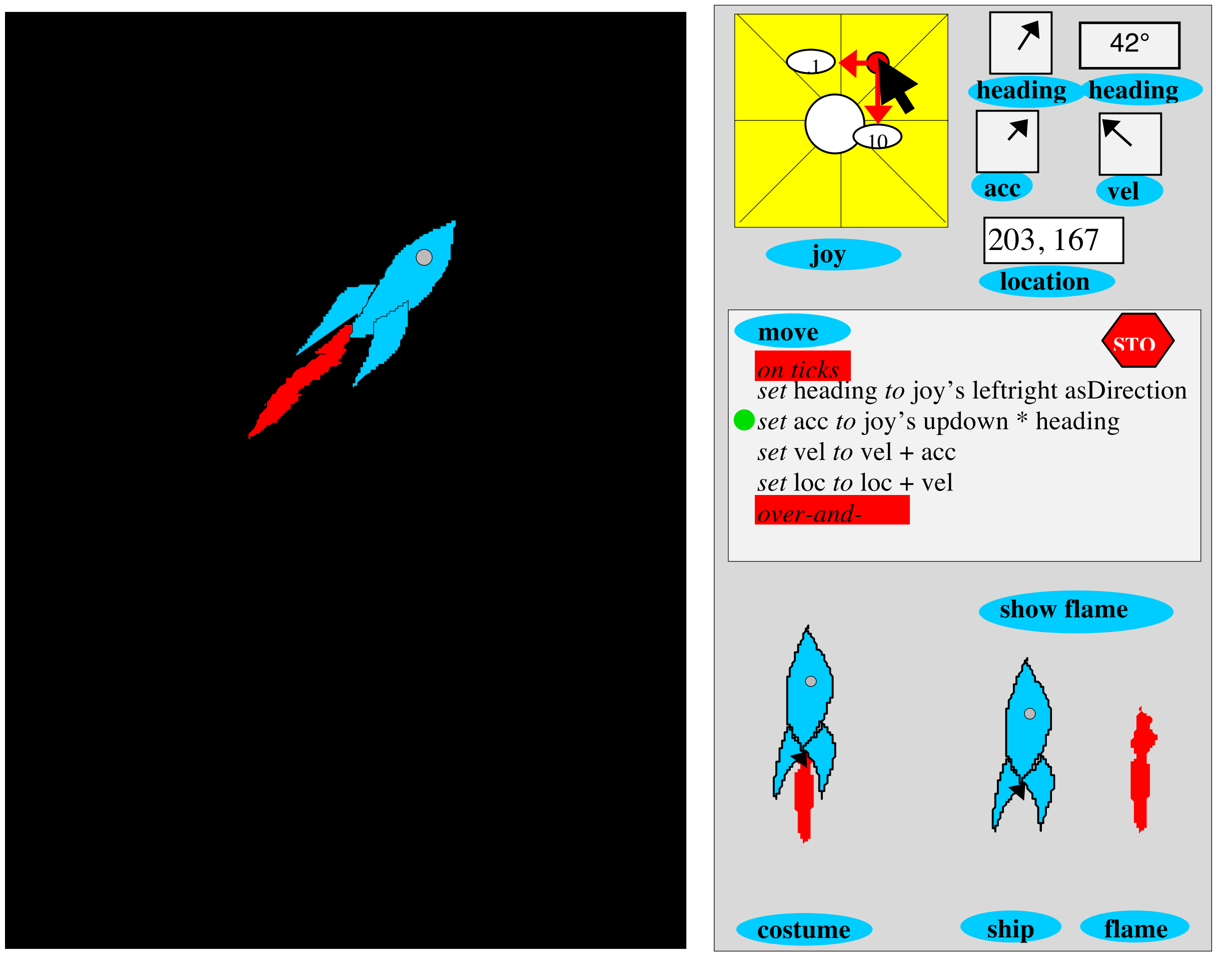


......MORE TO COME .....

mm

A More Elaborate Etoy and How It Is Developed

An example of a finished E-Toy is a dynamic spaceship that works just like a real one. We would see a spaceship floating in space and some controls which include a simulated joystick (this is another E-Toy), and a script for moving the spaceship.



```
move
on ticks
set heading to joy’s leftright asDirection
set acc to joy’s updown * heading
set vel to vel + acc
set loc to loc + vel
over-and-
```

The child’s goal is to make and play with a spaceship that behaves just like a real one. We need to scaffold this motivation so that the child is inspired to carry through on this project.

Our goal is to help the child understand a profound idea about accellerated motion in two dimensions—expressed in terms of a special computer version of vector calculus—by getting the child to write the four lines of the **move** script that animates the spaceship.

Let me out off to a later part of this note the explanation of how we get a child to create this E-Toy in Squeak. Let us first look at how someone who understands some of the profound ideas of physics would make this E-Toy, and then ask what has to be done to get these ideas in the range of a child.

E-Toy Construction Stage 1: Get a model of the idea going any which way

In this first stage, the E-Toy designer is allowed to go outside the boundaries of the child’s authoring system to get the idea model going if necessary. For example, we get the desired heading of the spaceship by adding a number gotten from the leftright outputs of the joystick. (We can prescale the joystick outputs so the child doesn’t have to worry about that task.) The script here is straightforward:

set heading *to* heading + joy leftright

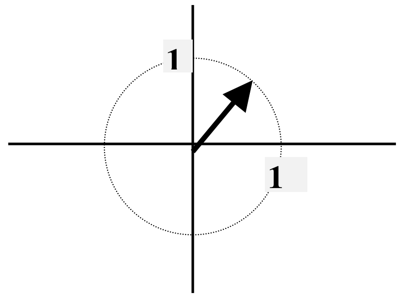
In other words, on each tick, get the new heading by adding the joystick leftright output to the existing heading. The heading will stay the same if the joystick is 0, will increase if the stick is to the right, and decrease otherwise. The resulting heading number is most understandable as the number of degrees of turn from some zero point—in this example, pointing straight up is the zero heading. In this model we assume that the heading of the spacecraft is controlled by gyros or by attitude jets in a way that allows us to reorient the ship without worrying about having to kill off angular momentum from a turn. (This is what current spacecraft do.)

Any accelleration that will be applied by the spaceship's rocket motors will be in the current heading direction. We want to get the accelleration of the spaceship from the updown output of the joystick. But accelleration in a two dimensional world can't be given by one number. Acceleration, like velocity, and location, needs two numbers to be specified. One way to give the two numbers is: a number for the magnitude of the accelleration, and a number for the direction of the accelleration. The E-toy designer would then use trig to convert the heading and magnitude numbers into a real accelleration vector that can be used to make velocities that can change the location of the spacecraft.

This will create a working model, but not one that can be made and understood by younger children...

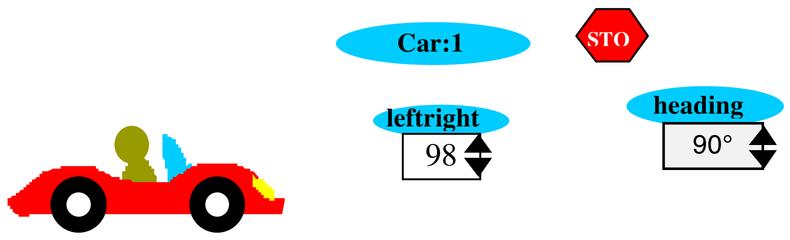
E-Toy Construction Stage 2: Make the model be what the child should make

What we really want to do is to criticize the Stage 1 model to find areas where the child's authoring system should be changed to make the scripting and concepts more clear. The fault in this example is that the system had a weak notion of what a direction should be. It is really more than just a number: it is at least a number that is tagged with the degree sign (°). In an object oriented system, the idea of a *direction* should be represented by a class called Direction, with methods that allow us to view its instances in felicitous ways—one of these ways should make it part of the world of vectors. One way to do this is to think of a *direction* as a vector that points in the right direction whose magnitude is 1.



set heading *to* heading + joy leftright

In other words, on each tick, get the new heading by adding the joystick leftright output to the existing heading. The heading will stay the same if the joystick is 0, will increase if the stick



Appendix 1: Current List Of Etoys and SimStories

Types of Etoys:
-- Just illustrates a point, but does not show code (like Ted's current wandering letters)
-- Vanilla etoy with visible editable code, explores one idea.
-- SimStory is a longer form which might enclude several projects
-- Parent's version of the above two: has a wrapping of additional explanation and tips

Etoys

* • how variation helps camouflage
* • how big and small can numbers get?
* • how long and short can time get?
* • motion -- via animation scripts (multiple etoys here)
* • motion -- via experiments (truck rolling down slope -> computer model) (multiple etoys here)
* • make a cannon, shoot the monkey, etc.
* • make cars you can drive
* • make spaceships, suns, planets
* • motion -- over sound samples
* • make an echo machine
* • make new timbres and play music with them
* • record sounds and play with them
* • make a song and orchestrate it
* • motion -- over images: magnification, distortion, etc.
* • how smells propagate
* • feedback (find the light, etc.)
* • build a thermostat
* • following a scent
* • make a simple painting system
* • learn to draw a ... : face, animal, cartoon character, etc.
* • make an animated flying bird
* • make a walking Mickey
* • etc.
* • "Felix the cat" type stuff/Klik 'n' Play with simple scripting
* • make an elastic wall bouncer
* • followers, ropes, etc.
* • Ted's Wandering Letters -- etoy version: exploring what it can do with simple scripts
* • differential circle
* • "Joe"
* • commander Pen, turtles, etc.
* • make subactivities of BUSYTOWN (using Disney backdrop).
-- The SeeSaw one is good for an etoy
* • make a clock from logic (doorbell) type
* • make a simple wall following robot using logic gates and a thruster (pre Rocky's Boots)
* • Make your own Web Pages
* • areas and volumes
• make a PONG
• make a Breakout game
• make a maze solver
• "monkey and bananas" solver
• seasons
• how high is your school, your building?
• how far away is the moon? How can we tell? Triangulation, etc.
• what shape is the earth? How can we tell?
• How big is the earth? How can we tell?
• make a clock from gears
• make the game of life, and other simpler such games
• make a biomorph

• what is a prism and how does it work?
• make your own "lenses"
• make a pinhole camera & a simulation of how it works
• make an eye
• differential forms (circles, etc.)
• What is air? Does air have weight? etc.
• what is the wind? Where does it come from?
• How do kites fly? Planes? Birds? (Differential pressure, etc.)
• How does a rocket work?
• clouds
• what happens when water boils and freezes?
• How long till my Birthday? (and the birthdays of my family and friends? Warning bells, etc.)
• make a one cylinder gas engine
• make a one cylinder steam engine
• day and night. How can we tell the earth is spinning and not the entire universe?

SimStories

* • Ted's Wandering Letters -- how to make it
* • car chase game
* • spacewar game
• make a simple PAC MAN game
• make a bowling game
* • make a simple "Defender" game
* • How to get out of a maze
* • make a "BIZZYDIZ" (Busytown with Disney characters and themes).
* • salmon swimming upstream
* • ants
• termites
• slime molds
* • epidemic
* • forest fire
• traffic
• binary sorting (gives rise to a bell curve)
* • gas expansion
• buoyancy. Start with sand and a heavy object. Where do upward forces come from?
* • predators and prey -- foxes and rabbits, rabbits and grass
• pond
* • Can Monkeys Type a Line of Shakespeare?
* • the 9 dots puzzle
* • Make a biomorph system
• how genetics works (improve your biomorphs)
• make a biobehaviour system (with genes for actions, etc.)
* • How to draw
* • How to animate
* • How to script
• how to compose
• how harmony works
• how counterpoint works
• how to arrange music
* • make a synthesizer
* • make a Wumpus game
* • how to make and control things with digital logic, circuits, etc.
* • Rocky's Boots
* • make a TIC TAC TOE player
• make a MUD
• ELIZA

- Parry
- make an email reader
- make an image transformer
- make simple animals
- hopping around (1D and 2D differential motion) -> animations
- airplanes and wind
- addition and subtraction "slide rule" -- w/ magnifying glass to see interesting part
- comeasurability of different measures (fractions)
- evolve an eye
- make your own SIMCITY
- * feelings coach (a la John Steinmetz and Tim Gallwey)
- the seasons
- the phases of the moon
- rain/water cycle
- what does is mean for something to dissolve (or not) in water?
- pendulum simulation

Appendix 2: The Squeak “Idea Processor”

In Computerese

The Squeak system is a small, efficient, completely reflective, and late-bound object-oriented environment for novice to professional authoring, implemented as its own multiplatform operating system from the "bits on up". It can run (a) completely standalone, (b) as an application on Macs, Windows, Suns, etc., or (c) as a plug-in or helper to other systems, such as Netscape and Internet Explorer.

Squeak is very small; in the extreme case less than 1/2 a megabyte each of ROM & RAM has to provided in order to run Squeak (in comparison: a full VGA screen color picture is one megabyte). It can run on Macs and PCs and workstations, but also can run on games machines, pocket organizers, and even television sets. Though Squeak can stand alone, it is at its best when dynamically combined with the Internet.

Squeak Press Release

Squeak is a new way for children of all ages to make *ideas* come to life on computers and networks. The range of these *ideas* comprehends the fairest part of human yearnings. Squeak can reach down into the earliest years of childhood and out to the farthest needs of professionals. Like English and mathematics, which can be used by children for their purposes and by Shakespeares and Newtons for theirs, Squeak has the happy facility of allowing simple ideas to be simple, and complex ideas to be possible.

Like a word processor, which serves authors as diverse as children, adults, and professionals for making "word-things", Squeak is an "idea processor"—for authoring anything a computer can make. Not just word-ideas and spreadsheet-ideas, and image-ideas, and sound-ideas, but process-ideas, simulation-ideas, and a vast range of other abstract-ideas, as well.

In order to make its way in the world, Squeak's *ideas* have to be of high quality (as good or better as the best things made on computers any other way), they have to safely inhabit every kind of computer, they have to work with non-Squeak kinds of things, *and* most important, they have to be made as happily and easily as possible by the widest range of users, especially children.

The Three Most Important Things

Probably the three most important things to understand about Squeak at this point are (a) that it is a "development platform for development platforms", (b) that it is its own operating system, and (c) that our special interest lies in designing and building ever better and more useful end-user authoring systems for children, parents, teachers, etc.

One implication of (a, b) is that there are many potential areas in which Squeak can help the Disney organization. These include new ways

• to create media on the Internet and CDROM

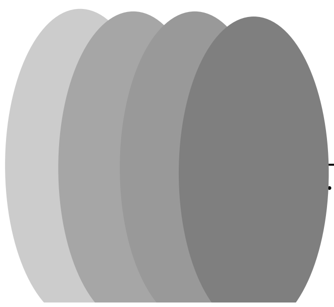
• to make special inhouse development tools (including rapid prototyping of many parts of Disney's processes)

Another implication of (a) is technical: unlike most development tools, which are chosen for their direct features, Squeak's main strength lies in its *potential to be shaped into what is needed at the time it is needed.* This is a very subtle and important trait that it shares with few existing systems.

A further implication of (a, b) is that Squeak can be quickly put onto any platform, regardless of whether it has decent (or even any) software at all.

The Structure Of Squeak

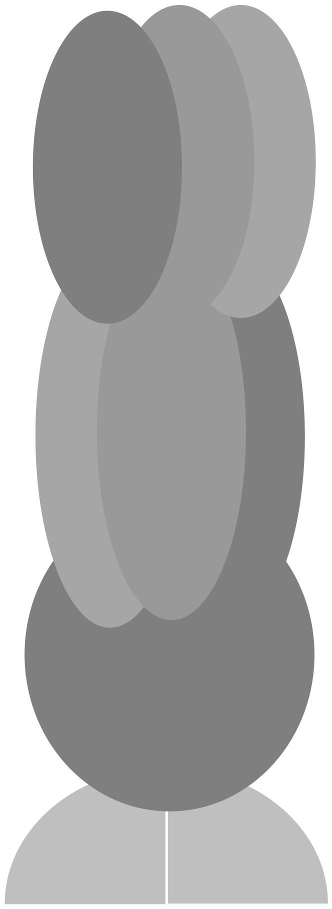
Squeak is a “sea of cooperating objects” with a vast range of services—from numbers to large scale simulations. One way to group these meaningfully is as the end-user might see them—the theater provides a helpful metaphor here. The images are overlapped to indicate that the boundaries are fluid between the layers—there are no walls to get through between them—all of the doors in the system are open, yet all objects are protected.



Stage/Show (Applications, Content, etc.)

At the nearest to the end user—at the “top” of the system—are objects that serve as tools and media. We call this the “content” or “stage” or “show” area.

Here the user is interacting with mostly finished media. For our purposes in this note, these would all be Etoys or SimStories. However, "show objects" could have been created with a wide variety of authoring environments at the next level.



Authoring, Producing, Costumes, Props

Here is where productions are created. There is quite a bit of overlap with the Stage/Show area because many of the show objects—such as Etoys—will provide considerable access to authoring tools as part of their content. Any number of different authoring environments can be established at this level—some might be aimed at children and novice end-users, others might be set up for professionals. The objects made here are mostly incremental changes to what exists in the prop and costume rooms, etc.

Authoring Of Authoring Environments

This environment is a heavy duty "machine shop" where fundamental new objects can be made from scratch for use upwards, and it has a special area that contains a working model of the next level down—"Physics of the World" (what computer people sometimes call a Virtual Machine). If changes need to be made to the virtual machine, they are made here, and then a special process converts the model to a new virtual machine and installs it automatically.

Physics of the World (Virtual Machine)

This contains the fundamental processes that run Squeak, including the Squeak interpreter, the graphics and sound primitives, the storage and other resource managers, etc. This and the level above are complete enough to constitute a standalone operating system, and it is quite possible to run Squeak on a bare machine by translating the VM simulator in the level above to the new machine's codes to produce this VM in terms of the new machine's codes.

Fundamental Particles (Computer Hardware)

This is the hardware of a computer. Many hardware designs will not permit any real overlap with the virtual machine; those that do (such as 2nd level microcode, etc.) can be used to advantage.