Squeak Etoys Authoring & Media
by Alan Kay

Summary of Squeak Etoys
Squeak Etoys (www.squeakland.org) was inspired by LOGO, PARC Smalltalk, Hypercard, and starLOGO. It is a media-rich, authoring environment with a simple powerful scripted object model for many kinds of objects created by end-users that runs on many platforms and is free and open source. It includes 2D and 3D graphics, images, text, particles, presentations, web-pages, videos, sound and MIDI, etc. It includes the ability to share desktops with other Etoy users in real-time, so many forms of immersive mentoring and play can be done over the Internet. It is multilingual, runs on over 20 platforms bit-identically and has been successfully used in USA, Europe, South America, Japan, Korea, India, Nepal, and elsewhere. See companion note Squeak, Children & Learning for more information on use by children.

Basic Idea
Provide and extend most personal computer needs via simple powerful user interface that “runs everywhere on everything” and allows end-users “authoring and access to all things”.

Examples
First we will take a look at some of the media in Squeak Etoys, and then go deeper to find that they are all made from the same kind of simple powerful end-user object, and scripted the same way.

All Objects Can Be Made And Used Everywhere
Everything in Squeak is an object, there are lots of already supplied useful objects, and end-users can make the same kinds of objects.

Integrated Objects Instead Of “Applications”
Separate applications are an old 60s idea which force a mode of authoring that is limited to what the application can do, and this makes new ideas by end-users difficult to fit in. For example, most Windows applications are not good for presentations, so Windows users will often do work in powerful applications (such as a spreadsheet or video editing tool) and then copy images to a weaker presentation tool. Looking at it from this point of view, there is only one “application” in Squeak Etoys – itself – and all old and new things are created, manipulated and presented there.

The creations of the end-users are the center of focus
As we will see, in contrast to most current systems which crowd the desktop with hundreds of buttons and options, the Squeak approach is to give the user a large place to work and make things, with a much larger set of resources at hand, but not intruding on what’s being made and done.
There Are Lots Of Resources “Backstage”
The end-user is usually engaged in making things, ranging from simple presentations of text and images to complex simulations. All of these are done on the desktop/project “page”. To avoid cluttering up the work/play area with resources and functions, Squeak instead uses “flaps”, which are hidden windows that are invoked by pressing on a tab. For example, “Supplies” is a flap that contains many useful objects to make things with, and “Navigator” contains useful ways to move about the Squeak worlds, find and publish projects, etc. The end-user can make as many flaps as needed, and these can be shared or just local to a particular project. They can be invisible for presentations or made visible to aid constructions.

Movement player can play MPEG and JPEG Movies

MP3 and other sound format player

The 2D environment is actually inside of a 3D environment that is mapped onto the Internet. The object in the lower right is a portal to another 3D world. See http://opencroquet.org for more information and “Squeak Etoys” to see how a dynamic model of a bridge is made.
Multiple Projects/Desktops
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 (But Delayed)
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.
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 ...

... make a script

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

The tiles fly into the destination when dropped.

The script can be started “ticking” by clicking on the clock

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 ...

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

... and write the same script we wrote for the car

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!
Everything Is Constructible, Changeable, Deconstructable

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.

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 and drag it out to start a script.

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

We drop it on “dot” to get:

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.

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

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..
Movies and Videos Are Just Animations
Now we should realize that we have also made the guts of a movie player because we can just drop frames from the movie into a holder, write the two line script and it will play them!

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 powerful) 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:

Sound Synthesis Is Just An Animation
Now let’s look at a different sequence in a holder. With the sound recorder we record a tone and see it is a sequence of bars, whose height indicates the sound pressure at that time.

One of the objects supplied with Squeak looks like a loudspeaker and if we move this object, the physical loudspeaker will move. We write a little script that is like the animation scripts, but instead of doing a “looks like” we will move the loudspeaker according to the height of the current bar, and we hear the tone! If we change the “increase by” to 2, we will hear the tone an octave above, and if we change it to 1.5 we will hear the tone a fifth above. We have just made a real-time synthesizer!

Squeak Allows End-Users To Access The Underlying Simplicity Of Many Important Ideas