Viewpoints Research Institute

Etoys for One Laptop Per Child

Bert Freudenberg, Yoshiki Ohshima, Scott Wallace

Published in the Proceedings of the
Seventh Annual International Conference
on Creating, Computing, Connecting and
Collaborating through Computing, Kyoto
University, Kyoto, Japan, January 2009

VPRI Technical Report TR-2009-001

Viewpoints Research Institute, 1209 Grand Central Avenue, Glendale, CA 91201 t: (818) 332-3001 f: (818) 244-9761

Etoys for One Laptop Per Child

Bert Freudenberg
bert@freudenbergs.de

Yoshiki Ohshima
yoshiki@vpri.org

Scott Wallace
scott.wallace@vpri.org

Viewpoints Research Institute
1209 Grand Central Ave.
Glendale, CA 91201

Abstract

We present an overview of the "OLPC Etoys" system, describe the intensive two-year development effort that produced the system, and discuss lessons learned. OLPC Etoys is an end-user authoring system for children, which was chosen to be distributed with the OLPC XO laptops at an early stage of the OLPC project.

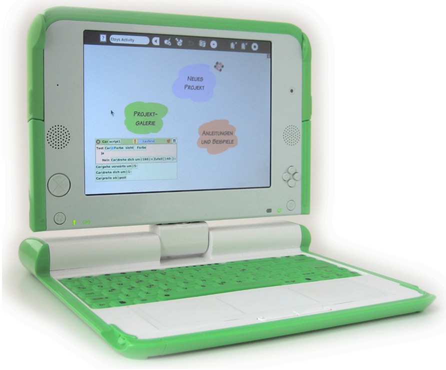
Since we planned to derive OLPC Etoys by evolving an existing, mature system ("Squeakland"), it was expected to be a relatively straightforward undertaking. However, the OLPC XO platform's special hardware characteristics, the evolution of the Sugar software stack, and the fundamentally international and multilingual nature of the project, all conspired to make the development effort challenging.

Over the two-year course of the project, we successfully kept up with the challenges, and delivered usable Etoys systems for every OLPC release. We steadily improved the UI, added a few high-leverage features, and fixed bugs, with a small and widely-distributed team and with help from the community.

1. Introduction

In this paper, the short but intensive history of Etoys in the One Laptop Per Child (OLPC) project is described. From the early days of the OLPC project, when it was still only an idea being discussed at the MIT Media Lab in 2004, the Viewpoints Research Institute (VPRI) was involved because our group, led by Alan Kay, had a long history of researching and developing the concept of the portable educational computer, and, more generally, of educational software, dating back as far as the seminal 1968 "Dynabook" design [1].

At that time, in summer 2004, VPRI had a well-recognized educational software package called "Squeak Etoys" which had been freely available for some years [2] [3]. However, VPRI defines itself as a research organization, and since the development work was already "done" from a research point of view, VPRI was looking forward to new design and architecture ideas for the "next generation" Etoys-like system at around the time the OLPC project was being formulated.



```
Car script1   Laufend
Test Car Farbe sieht Farbe
  Ja
  Nein Car drehe dich um 160 + Zufall(40)
Car gehe vorwärts um 5
Car drehe dich um 1
Car pralle ab psst
```

Figure 1. Etoys Activity's start up screen.

As the saying goes, "the timing is always bad." On one hand, we felt a strong impulse to apply our ideas from years of experience with Squeak Etoys into a brand new system, built from the ground up from first principles – something better and cleaner and purer. On the other hand, the time window available for producing an XO-ready version of Etoys was very short; the announced launch date was less than two years out, and we realized that it would be a high risk prospect to try to deliver a brand-new, comparatively untested piece of software in such a short time-frame for a platform (itself brand new) that would be produced in massive quantity.

There was some internal discussion as to which way we should go. The key decider was the ongoing presence of numerous uncertainties around the OLPC project; we concluded that it was best for us not to be adding more uncertainties from our end by trying to create a whole new system; thus, by the time software effort got started in earnest, VPRI had decided to take the existing Etoys and adjust it to the OLPC platform.

With this conservative approach, the task seemed straightforward. However, of course, the effort took many more person-hours than anybody had imagined. OLPC was inventing new hardware (later named the "XO Laptop",) and performance requirements kept changing. OLPC was also inventing a new graphical user interface called "Sugar" that itself kept undergoing considerable change. Since buyers of XO's would be primarily third world countries' governments, localizing software became more important, and localization efforts consumed a great deal of our efforts. As the team made progress, it turned out that the task was not straightforward at all.

However, the team managed to release a continuous series of robust and usable systems, with constant improvement along the way. A big factor was help from the open-source community, and another was the malleability of the Squeak system on which Etoys is built [4].

The rest of this paper is organized as follows. Section 2 summarizes the Sugar environment. Section 3 describes the XO laptop hardware and the challenges it poses. In section 4, the participation of the international community is described. Section 5 and section 6 discuss some of the actual improvements and changes made to the Etoys system, and the accompanying materials created. Section 7 discusses the efforts in the matter of licensing. Section 8 summarizes the contributions and activities in the extended community of Etoys users and developers.

2. Sugar integration

In August 2005, there was a meeting between engineers from RedHat, who were going to work on the system software side of the OLPC project, and the Etoys team. At that time, the laptop was designed to have 128MB of memory, of which, we were told, 70MB would be reserved for the base system (later, the system code itself grew larger than 128MB, and the hardware was redesigned to have 256MB.) Over the following 12 months, the OLPC team produced an early prototype of a software stack, and external developers such as the Etoys team got various prototype hardware, including the AMD Geode GX developer's board and a board called "Rumba" that was closer in form factor to the final product. (The GX processor was later replaced with the faster AMD Geode LX processor.) The team experimented on each revision of prototype hardware using the existing Squeak Etoys to understand the system.

In September 2006 the Etoys team gathered for two weeks in Cambridge, Massachusetts, to visit the One Laptop Per Child headquarters. This marked the official beginning of a two-year programming effort that resulted in the release of a new, much-improved Etoys version in September 2008.

In 2006, the available hardware consisted only of prototype boards, running Linux with a software stack similar to a regular Linux desktop, including an X11 display server. Etoys worked the first time we tried it on the brand-new screen prototype, which had just arrived from the factory in China.

But there was also much to do. Software development on Sugar, the graphical user-interface for the laptop, had just begun, and it would present many challenges over the following months to integrate Etoys well with this ever-moving target.

But getting a first proof-of-concept implementation running inside Sugar was accomplished in only a day. It used a wrapper implemented in Python like the rest of Sugar. This wrapper launched Etoys and redirected its window into a Sugar window, re-using the window-embedding technique originally developed for the Squeak web-browser plugin. Commands were passed from that wrapper into Etoys using a "pipe", but there was no back channel to access Sugar functions.

In June 2007 we retired the Python wrapper and made Etoys integrate with Sugar natively. Applications communicate with Sugar using the D-Bus message bus system [5]. Thanks to a new DBusPlugin we could now directly integrate Etoys with Sugar. This eliminated the overhead of running a Python interpreter that took about 10MB of memory, and also reduced launching time of Etoys from tens of seconds to mere seconds. More importantly, the change gave us direct access to the Sugar APIs from within the Etoys system.

Providing proper access to the "Journal" and the "DataStore" were big challenges. Sugar does not present the user with a traditional file system, but instead automatically records user actions in a database. Children do not have to load and save files manually; instead activities (i.e., Sugar applications) are required to store their state without user interaction into the storage abstraction called DataStore. Later an activity can be "resumed" from the Journal, which presents a chronological, searchable view of all previous activities.

Therefore, under Sugar Etoys needed to use the API for accessing the Journal and the DataStore to save and load projects. Since that API kept changing rapidly and often in incompatible ways, we had to track the system changes more or less continuously, and often had to adjust Etoys to conform to API changes, sometimes at extremely short notice. Etoys was the first major non-Python activity on the

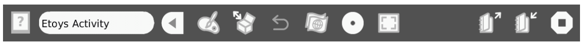


Figure 2. The Navigation Bar.
The navigation bar has Help, Title, Back, Paint, Supplies, Undo, Language, Share, Scale, Find, Keep, and Stop buttons.

XO, and as such was the first to use the low-level D-Bus API (whereas all the other prominent activities were implemented in Python using high-level wrappers). At times we even had to reverse-engineer the API, but we documented our findings, thus paving the way for other non-Python activity developers [6].

Consequently, Etoys can now follow Sugar's persistence conventions completely. Etoys automatically saves the current project when exiting. There also is a "keep" button that keeps a copy of the project. The "find" button does not open a file dialog but instead allows the user to choose and insert graphics, sounds, and other content from the Journal. Another way to exchange data with other activities is via drag-and-drop, and yet another is using the clipboard, which has been extended to handle pictures, formatted text, and other data.

Another important feature conceived of by the Sugar designers was the notion of pervasive and integrated collaboration support. XO machines are designed to announce their presence on the network automatically, so it is much simpler than on a traditional machine to connect to someone else. Etoys already had its own screen sharing and data sharing mechanisms, but they worked in ways that required users to provide a prospective peer's IP address manually, so that, in practice, it was not conveniently usable by children.

Leveraging off automatic presence announcements sent by Sugar, Etoys quickly gained the ability to establish sharing with simple steps. This was done during a visit to the OLPC headquarter in May 2007. Later, as Sugar got more sophisticated, it became possible for an Etoys session to be shared publicly by simply clicking the Share button in the navigator bar, or privately by inviting someone from Sugar's neighborhood view. In either case, the remote parties can join the activity by clicking the Etoys icon that appears on their machines. For each joining "buddy" a badge is created in a flap. Etoys objects (even scripted ones) can be dropped onto a buddy's badge and then get copied over to the remote machine. This allows a project to be collaboratively assembled. Such collaboration works equally well whether or not a server is involved, thanks to the Telepathy framework [7] that Etoys uses to create "tubes" for remote communication.

Sugar activities are normally packaged as self-contained "bundles," which are zipped directories bearing a ".xo" file extension. But to facilitate writing Squeak-based activities that do not have to include a full Squeak Virtual Machine and object image, OLPC decided to include those in the base system. The Etoys bundle only contains meta-data and a tiny start script that runs the pre-installed VM and image. In fact, we provide a function to create an XO bundle from an Etoys project file. This creates a skeleton bundle that can be edited to make a Sugar activity, even on non-OLPC machines.

To match the look of other Sugar activities, there is a new iconic navigator bar (See Figure 2). It contains a field to edit the project name and buttons to access the most frequently used functions. Advanced functions can be accessed using the XO's "view-source" key which brings up a menu to access many Squeak features (available as "Cmd-," on other machines, too.)

3. XO hardware support

Designed as a dedicated learning machine, the XO-1 laptop has some unique hardware features that demand special attention in the software, too.

The most conspicuous feature is the high-resolution 1200x900 pixel display at a density of 200 pixels per inch. This is twice the resolution of a regular laptop, meaning that things will appear at only half the physical size if they occupy the same number of pixels. It also employs a special color-scheme that does not rely on sub-pixel color components; this feature makes casual use of single-pixel-width lines unadvisable. Thus, for example, the sizes of the nibs available in the Etoys painting system needed to be adjusted.

While the pixel-resolution is high, the processor it uses is a modest Geode LX running at 433MHz. Software needed to take the discrepancy into account.

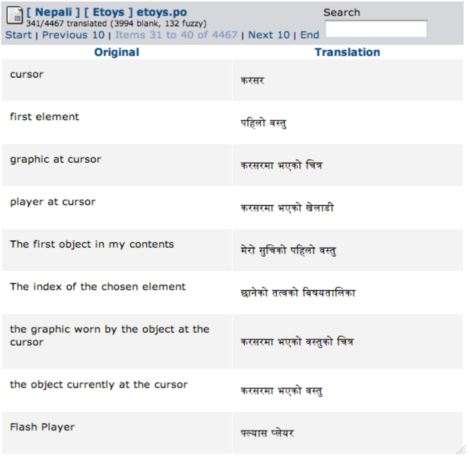
Another unique feature is the XO's microphone jack. It allows not only microphones but also sensors such as a photo resistor to be plugged in. The analog-digital convertor has to be switched to a DC mode for this sensor support (normal microphone input filters out the DC component,) which has been added as a primitive.

The built-in camera is supported by a new plugin; using it, users can record both still-frames and image sequences directly in Etoys.

For video playback, the Geode LX processor has special-purpose scaling and color-conversion hardware, which can be accessed using the new "GStreamer" plugin.

The XO's mesh network is anticipated to use IPv6 addressing, so Squeak's network support was extended to allow other socket types than IPv4.

One important design characteristic of the XO is low power consumption, so we added a feature to suspend Etoys while it is not running in the foreground.



[Nepali] [Etoys] etoys.po
341/4467 translated (3994 blank, 132 fuzzy)
Start | Previous 10 | Items 31 to 40 of 4467 | Next 10 | End

Original	Translation
cursor	करसर
first element	पहिलो वस्तु
graphic at cursor	करसरमा भएको चित्र
player at cursor	करसरमा भएको खेलाडी
The first object in my contents	मेरो सुचिको पहिलो वस्तु
The index of the chosen element	छानेको तत्वको बिषयतालिका
the graphic worn by the object at the cursor	करसरमा भएको वस्तुको चित्र
the object currently at the cursor	करसरमा भएको वस्तु
Flash Player	फ्ल्यास प्लेयर

Figure 3. Pootle, the web-based translation system.

Table 1. Coverage of Translation.

Language	Coverage	Language	Coverage
Turkish	100%	Italian	4%
German	99%	Nepali	2%
Mongolian	94%	Russian	2%
French	90%	Pashto	1%
Spanish	80%	Bulgarian	1%
Kreyol	65%	Marathi	1%
Japanese	60%	Dutch	1%
Portuguese	43%	Telugu	0%
Dari	41%	Arabic	0%
Sinhala	40%	Chinese	0%
Greek	35%	Romanian	0%
Swedish	9%	Bengali	0%
Urdu	6%	Korean	0%

4. Internationalization

In October 2006, an effort was started to improve Etoys translations. The tiles, system messages, dialogs, and other UI elements were originally coded to show English phrases, and the translation mechanism uses a dictionary lookup to provide the UI in different natural languages. We replaced the old translation system with a new one provided by a community member based on "gettext" [8], which is the pervasive standard for localization in the open source community. Translatable strings are marked in the image using #translated sends; these are extracted to a "pot"-file, which serves as template for the "po" files containing translated phrases.

A world-wide community of translators uses the "Pootle" web-based translation system, which allows simple online or off-line editing of translations (see Figure 3). There are more than 4000 phrases to be translated in Etoys alone. As of September 2008 translation has been started to Arabic, Bengali, Bulgarian, Chinese, Dari, Dutch, French, German, Greek, Italian, Japanese, Korean, Kreyol, Marathi, Mongolian, Nepali, Pashto, Portuguese, Romanian, Russian, Sinhala, Spanish, Swedish, Telugu, Turkish, and Urdu.

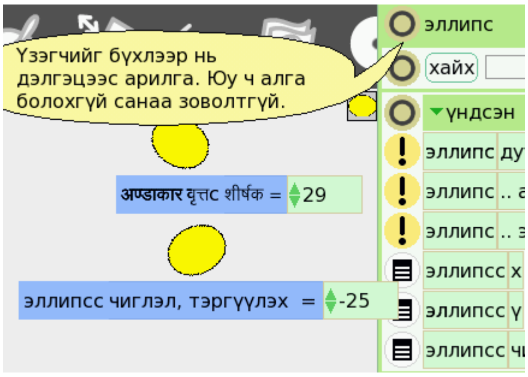
Table 1 shows the coverage of translated phrases in these languages, sorted by completeness of coverage as of late 2008. There are some interesting facts:

- Turkish went all the way, and Mongolian was almost there. Both countries are participating in small pilot programs and their members continue to give close attention to the Pootle server, noticing and translating new phrases quickly. Kreyol is in a similar situation.
- The "old" group. Portuguese, French, German, Spanish, and Japanese, all already had good coverage from the old Squeakland era. The German community was very active in proceeding to translate the whole system, but the effort to extend translations in the other languages in this group to full coverage has not yet gotten full traction. Nevertheless, community members from these regions are active and helping with coding, testing, and other work.
- While no pilot test is going in Greece yet, some active participants are trying to make it happen. The translation coverage seems to confirm that, as exemplified by the Dari and Sinhala translations targeted at pilot countries.
- Some languages have very small coverage. Typically, one or two people might start a translation effort, but perhaps due to the hard-to-use interface of Pootle, and perhaps due to the overwhelming number of phrases, small groups tend to stall.

When starting up, Etoys switches to the system locale, which it infers using the LocalePlugin. All the toolbars, menus etc. get translated.

Many languages use scripts that cannot be displayed correctly using traditional Squeak bitmapped fonts, nor even using Squeak's TrueType font renderer. For those languages we now use a text renderer based on the open-source Pango library [9] which can properly show nearly every script in use. Figure 4 shows a screenshot with a watcher in Nepali and rest of UI in Mongolian. An unfortunate side effect is that now the display depends on the fonts installed in the system, whereas previously projects worked bit-identically across all platforms and all client machines. We do not know how to solve that issue yet. Also, work remains to be done to support right-to-left languages.

Another area we have not really tackled yet is translations of project content and help guides. When a project



Үзэгчийг бүхлээр нь дэлгэцээс арилга. Юу ч алга болохгүй санаа зоволтгүй.

Figure 4. Screen shot of Pango rendered text.

is loaded, it is partially changed to conform to the current locale, so that for example the tiles and controls in scriptors and viewers are translated. But only very rudimentary supports exist for translating user-added text.

A user can mark a piece of text "translatable" using an item in its halo menu; basic text objects obtained from the supplies bin are already marked as "translatable," and others can be so marked at user discretion. Text marked as "translatable" can be edited as usual, and when the language is switched (using the flag icon in the navigator bar) the contents will automatically be changed to the version that item had in that locale previously. The problem with this simplistic approach is that the layout in a different language might not work as well as it did in the original. Also, this is not yet integrated with the gettext translation engine, so projects cannot yet be edited from Pootle.

5. Etoys improvements

We added new features, improved expressiveness, made adaptions for unique features of the XO hardware, and fixed numerous bugs over the course of development of the Etoys system. The Squeakland Squeak system, which Etoys is based on, had been used widely around the world for quite a few years, over the course of which much useful and interesting feedback had been received. Since we realized that the new Etoys system would be even more widely used, we made several significant enhancements and adjustments. In this section, we discuss some of the new features and other changes.

5.1. Expressiveness

The original Squeakland system was intended for younger children whose computational needs would be satisfied by the basic arithmetical operators. Since users of the new Etoys version on the XO would include older children, and indeed users of all ages, we decided to add a "function" tile and to add visible parentheses to make plain precisely what the arguments to a given function call are; around 25 built-in functions are provided, such as square root, exp, sin, log, etc., which the user can select from a pop-up menu, and the same mechanism also provides a useful generalization of the pre-existing but much less flexible random tile, whose argument can now be an arbitrary expression whereas formerly it had had to be a constant. Also added, using the same mechanism, is a plain "parentheses" function, allowing the user explicitly to specify an order of evaluation.

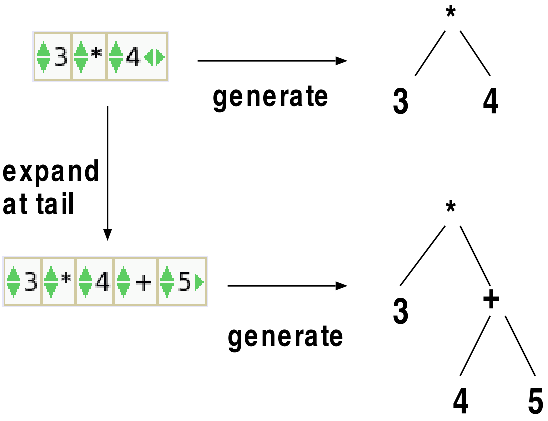


Figure 5. The widget tree representation of an expression.

Furthermore, and more significantly, we were able to remedy a major shortcoming of earlier Etoys versions, having to do with operator precedence and order of evaluation of arithmetical tile expressions.

In previous versions of Squeakland Etoys, operator precedence in scripts was always right-to-left, without regard to traditional "strength" of operators; e.g., a group of tiles that reads: "$3 * 4 + 5$" was interpreted as "$3 * (4 + 5)$" and resulted in 27, instead of 17 that a fourth grade child, or indeed anyone, would expect.

This was due to the internal representation of tiles. A group of tiles is a tree of graphical widgets. Expression expansion, only permitted at the tail, is done by replacing the last element of the expression with a sub-tree. For example, in Figure 5, the widget that represents "4" is replaced with a tree of widgets that represents "$4 + 5$". When the code generator generated executable code from the widget tree, the nested structure was directly converted to nested expressions, thus resulting in the unconventional precedence.

Regardless of the internal structures involved, we wanted the actual operator precedence used in evaluating arithmetical expressions in scripts to conform to conventional expectations, e.g. $*$ and $/$ must be stronger than $+$ and $-$. To make this happen, the code generator now makes an extra pass to convert the parse tree to another parse tree in a manner of operator-precedence parsing with a stack.

Another addition to improve the expressiveness is the repeat tile. A repeat tile takes the repetition count and a block. When executed, the tile repeatedly evaluates the block the number of times specified by the count, which may itself be a tile expression.

5.2. Design Changes

The primary target platform of the Etoys system is the OLPC XO, which has a peculiar combination of high pixel count, yet small display (i.e., high dot-per-inch number around 200), and a relatively slow processor (see Section 3).

To adapt to the characteristics of the XO platform, we settled on a few UI design principles:

Larger UI elements Because of the high DPI and younger audience, basic UI elements such as text labels and buttons should be larger on screen.

Cleaner look has performance benefit Rounding the corners on widgets and tools, and using gradient backgrounds, both of which were common practice in earlier versions of Etoys, both came with performance penalties. Flat color and right angle corners also gives a crispy feel.

Simpler Menus Over the many years of development of the Squeakland Etoys system, feature creep temptations were often given in to, by many different hands. Many features, some of them arguably unimportant or confusing, were buried in a range of obscure menus. Even the icons to pop up menus were different in different parts of the user-interface. Menus should be simple and consistent.

Following these principles, larger font sizes were used, halo handles were enlarged, and the UI layout was generally adapted to suit the increased resolution. But simply making everything proportionately larger was not always a perfect strategy because of the need for screen real estate to be adequately apportioned out among user-interface elements. One particularly satisfactory innovation was to make it such that when the mouse cursor hovers over the arrows in a value tile, enlarged versions of the arrows pop up under the mouse pointer, to allow more accurate clicking. A new primitive was added to support larger, anti-aliased mouse pointers (which formerly were limited to 16x16 pixels in black-and-white.) The looks of viewers and scriptors were flattened so that they do not use gradient and rounded corners.

A notable feature we added to overcome the screen resolution issue was a screen-scaling feature that can resize the Etoys display. This is especially useful on non-XO machines. With scaling enabled, project authors can create projects in 1200x900 resolution regardless of the actual display size. Furthermore, if a user on a non-XO machine resizes the Etoys window to about 6x4.5 inches, the viewing angle of the XO display can be simulated.

5.3. New Objects

We added several new widgets, and also improved existing ones.

New widgets include a significantly improved particle system (called "Kedama") [10]. The particle system in Etoys lets the user program thousands of objects using the same tile-scripting interface that is used for normal scripting. The new particle system in the Etoys system utilizes homogeneous array arithmetic primitives with mask bits; e.g., primitives that can perform operations on selected elements. With such a primitive, a vector computation can be carried out in the primitive even when it is in a test tile. However, the data representation, which heavily relies on the floating point numbers, has a performance penalty on the XO's Geode LX processor.

"WorldStethoscope" [11] is an interface connecting Etoys to various physical sensors. It consists of a hardware part that converts digital or analog sensor output to an audio signal, and a software part that provides Etoys tiles for observing and manipulating the signal. An earlier version of WorldStethoscope only took input as an audio signal, but the version in the new Etoys system is enhanced so that it can utilize the direct analog input mode of the XO (see Section 3). And the XO's on-board camera input can be used in the Etoys system via the "VideoForLinux" library and Squeak bindings to it.

Other widgets were modified to take advantage of the XO hardware and the Linux platform. The SoundRecorder was completely rewritten, simplified, and can now compress the recorded sound with the Ogg codec. A VM interface to the "GStreamer" media framework [12] was added. The clipboard mechanism can copy and paste rich text, as well as other types of objects such as bitmaps, to and from Sugar. The Book widget is now equipped with the ability to revert pages. The EventRecorder mechanism is enhanced to be able to control the flow of events and edit them. Vertices of a polygon now can be manipulated via tiles.

6. Learning materials

Educational software can seldom have any meaningful impact unless accompanying teaching materials or other forms of guidance are available for users. Most previous use of Etoys had been done in classrooms, with knowledgeable teachers providing instruction and support. But many users of XO machines around the world, if they are to learn to use Etoys, will have to do so without the benefit of a teacher who already knows the system. So we undertook to provide a range of introductory materials, including tutorials, a help system, and a suite of examples, all of which are included, and readily navigated to, in the basic Etoys installation on every XO machine.

The Squeakland system used to open into a blank white screen when launched. The intention of the blank screen was that users should be encouraged to do "whatever they want" and start from scratch. It turned out, however, that many users had difficulty getting started; at a minimum, users needed to know to click on the paint brush button in the pop-up navigator bar to paint a new object, or to drag from an icon in the supplies flap to instantiate a new object. And there were no good examples readily available to play with. In reconfiguring the system for the XO, we addressed such shortcomings.

Also, we wanted users and learners to engage in deep, powerful ideas, especially in science and mathematics. Since children are not often going to discover profound ideas on their own, we felt that good guidance, both for children and for teachers, was necessary. Also, the Etoys user interface does require some explanation if the user would like to do more than just painting. A large variety of kinds of objects, and a large number of features, are available, but users are not likely to discover much of the advanced functionality until they develop a basic level of proficiency with the halos, menus, navigator bar, supplies bin, and a few other basic tools in the Etoys environment.

In the Etoys system, we decided to provide an appealing initial screen, a suite of examples, and a built-in library of "Quick Guides" for reference. When Etoys is launched, the initial screen looks like Figure 1. There is a car object moving around, its script is visible, and there are three buttons to open example Etoys projects, to open tutorials and a self-repeating demo, and most importantly, to create a new blank project.

There are about two dozen example projects provided, organized roughly in order of progressive complexity. Ten of them are taken from an actual curriculum used in classrooms [3], and others illustrate math and science ideas.

The tutorials are presented in the manner of "puzzle-solving" games, to engage children. We received much very favorable feedback on these tutorials from users.

The Quick Guides, always immediately available via a single click on the "?" icon at the left edge of the navigator bar, provide concise reference information for dozens of features of the system.

These on-computer materials are fun and engaging for the users, but they do not explain the really powerful ideas behind them. A draft of a paper discussing some of these ideas is available [13].

7. Licensing

Before Alan Kay's team left Apple Computer in 1996, they released Squeak to the general public. The license (the Squeak License [14]) was intended to keep Squeak free and open while at the same time allowing it to be used for proprietary work. The license attracted a world-wide community of developers who collaborated to improve and extend Squeak in a truly open-source fashion.

After the release of Squeak, open source licensing was formalized by various organizations in the late 90's, and some new standard definitions of Open-Source emerged. Unfortunately, the Squeak License had two clauses that made it incompatible with the new standards, which are exactly what the OLPC project is based on. So in 2006, VPRI approached Apple to re-license Squeak. Apple agreed and relicensed Squeak to the "Apple Public Source License" but before long that license was found to be flawed. Ultimately, Apple relicensed Squeak again under the "Apache Public License 2.0" which is an officially recognized free license.

However, that relicensing only covered the original 1996 release and by then 10 years of subsequent development had gone into Squeak. So VPRI decided to contact every contributor in the worldwide Squeak community for their permission to relicense their code. They were asked to relicense under the "MIT" license which is one of the simplest free software licenses and compatible with nearly every other license.

Over the two years of this effort, we collected relicensing agreements from more than 200 contributors, which covers over 99.4% of code in the system. Not every contributor could be reached however, and ultimately the remaining code that was not relicensed was removed or rewritten. As a result, a version of the system that contains only code under the Apache and MIT licenses is now available as "Etoys 4.0" from the VPRI website.

8 Community

One of the most rewarding experiences in this project has been – and continues to be – the interaction among a worldwide community of developers, users, translators, educators, and other interested parties. There have been literally hundreds of bug reports and suggestions for improvements filed on the bug tracking system, and the OLPC mailing list has been a lively and useful forum. There were even some developers contributing unsolicited bug fixes and enhancements.

The work is distributed and connected over the Internet, but at a few occasions, the developers and users have a time to meet up at some locations in the world. Notably one of the authors was invited by OLE Nepal and spent a fruitful two weeks with the developers of the Etoys-based "E-Paath" learning activity.

From the larger `squeak.org` community, many members were supportive and provide proactive help. There is even a developer who only reveals his pseudonym but who contributed a large part of the new translation framework.

9. Conclusions and Future Directions

This paper narrated the history of Etoys development for the OLPC platform. Adapting the system to the Sugar environment was an interesting challenge, since Sugar was a moving target, but we stayed in close communication with the Sugar developers, and managed to keep in sync with the changes. The hardware posed some interesting challenges, too, so we adapted Etoys to the special hardware features of the XO such as the high resolution display, direct analog input, and camera.

One of the biggest insights we had was to realize that a software product with the extent of Etoys cannot ever be "done". There are always some bugs, and there are always request for more features. Everybody shares an aspiration for cleaner code, but often this is not achievable. Also, some of the new requirements were hard to bring in after-the-fact; for example, the casual use of "camelCase" in the UI is arguably not good for children, but it was hard to remove; the user interface was not designed to accommodate languages written from right to left; and some attempts to optimize the project saving mechanism did not produce the expected outcomes. And we still have not yet been able to provide an efficient mechanism to support other Sugar activities written in Etoys such as "DrGeo II" and "Bots Inc."

The Etoys project continues, but there is some organizational change ahead. VPRI is creating a new organization, provisionally named the "Etoys Foundation," that will be spun off to assume responsibility for development and support of the Etoys system. The new foundation will look to get more educators as well as developers to participate in the world-wide Etoys effort. Meanwhile, VPRI will focus its research efforts on creating a worthy, and much more powerful, successor to Etoys.

Acknowledgments

We would like to thank the colleagues at VPRI and all the `squeak.org` community, especially Dan Ingalls and John Maloney, for making Squeak the malleable system allowing us to have fun while coding.

Also, for help with the OLPC XO hardware and Sugar software, we would like to thank Walter Bender, Jim Gettys, Samuel Klein, Michael Stone, C. Scott Ananian, Chris Ball, Marco Pesenti Gritti, Tomeu Vizoso, and Sayamindu Dasgupta.

References

[1] A. Kay and A. Goldberg, "Personal dynamic media," *IEEE Computer*, vol. 10, no. 3, pp. 31–41, 1977.

[2] A. Kay, K. Rose, D. Ingalls, T. Kaehler, J. Maloney, and S. Wallace, "Etoys & SimStories," February 1997, ImagiLearning Internal Document.

[3] B.J. Allen-Conn and K. Rose, *Powerful Ideas in the Classroom*. Viewpoints Research Institute, 2003.

[4] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay, "Back to the Future – The Story of Squeak, A Practical Smalltalk Written in Itself," in *Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 1997, pp. 318–326.

[5] "D-Bus message bus system," `http://dbus.freedesktop.org/`.

[6] "Sugar API," `http://wiki.laptop.org/go/Low-level_Activity_API`.

[7] "Telepathy: Flexible Communications Framework," `http://telepathy.freedesktop.org/`.

[8] "GNU gettext utilities," `http://www.gnu.org/software/gettext/`.

[9] "Pango," `http://www.pango.org/`.

[10] Y. Ohshima, "Kedama: A GUI-based Interactive Massively Parallel Particle Programming System," in *Visual Languages and Human Centric Computing*, 2005, pp. 91–98.

[11] K. Abe and T. Hayashi, "Squeak Trek - Adventures with World-Stethoscope," `http://www.ipa.go.jp/SPC/report/03fy-pro/mito/15-897d.pdf (excerpt from the IPA report in Japanese)`.

[12] "Gstreamer - open source multimedia framework," `http://gstreamer.freedesktop.org/`.

[13] A. Kay, "Children Learning by Doing: Squeak Etoys on the OLPC XO," Viewpoints Research Institute, Research Note RN-2007-006-a, 2007, `http://vpri.org/pdf/rn2007006a_olpc.pdf`.

[14] "The Squeak License," `http://squeak.org/SqueakLicense/`.