



SQUEAK / ETOYS

Reference Manual

Published : 2011-04-21

License : GPLv2+

Table of Contents

INTRODUCTION

1 INTRODUCTION	2
----------------	---

GETTING STARTED

2 GETTING STARTED	6
3 USER INTERFACE	13
4 COMMONTILES	34
5 OBJECTS	50

APPENDICES

6 License	94
7 AppendixEtoysFriendlyOff	100
8 Somebody Should Set The Title For This Chapter!	101
9 Appendix: Morph	102
10 AppendixMIMEtypes	104

INTRODUCTION

License: *The Etoys manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, **you agree** that your edits can be used under both the **GPL and MIT** licenses.*

1. Introduction

1.1 What is Etoys

Etoys is a highly engaging programming environment that allows students to create just about anything from interactive drawings and stories to models and games. With a "drag and drop" interface, it helps prevent syntax errors that can frustrate learners new to programming, while encouraging "what if" questioning. The projects that are possible are endless, Etoys can be used for simple drawings as well as for complex models of the physical world. While it was designed for 8-13 year olds, it has been used successfully by teachers at the high school and college level as well as by younger audiences with help from a parent or teacher.

To get an understanding of the Etoys environment, imagine you have written and wish to produce a play. The objects are the actors and Etoys allows you to easily program their actions. Etoys comes with ready-to-use objects in the Object Catalog (see A at the left on image below). Users can choose from a variety of text, graphic, multimedia, communication, and complex objects. The yellow ellipse object (see B at the bottom) is accompanied by a surrounding halo of menu items (the valuable balloon help is indicating the Viewer). One of the most useful menu items is a Viewer (see C on the right), which contains graphical programming tiles that can make the object perform a variety of actions. Some of the basic programming tiles available to the simple ellipse object can be seen in the Viewer. The object can make a variety of sounds, move forward by any amount, turn by any amount, move to any x or y position, or change its heading. There are several more menus of programming tiles that can be used to make each object perform further actions. The program to produce the play you are directing is started by dragging one of the tiles onto the workspace. A program that will move the yellow ellipse forward by 5 appears to the left of the yellow ellipse (see D at the bottom). Once you start, your project is limited only by your imagination. A very important aspect of the Etoys environment is the ability of students to author their own projects or build on projects created by others. This opens the door for a more authentic approach to learning.



With Etoys, children can draw their own sketches then bring them to life by writing "scripts" that tell the sketches what to do. Children can then put sketches and text in digital books with multiple pages, allowing them to create interactive stories to share with the world. Such sustained creative projects foster a sense of ownership and teach children the value of iterative refinement as they improve their stories over time.

Children can use Etoys to make their own models, stories, and games, which keeps them engaged because it's a lot of fun. But Etoys isn't just child's play. It's a highly effective way to teach math, science, and language arts, although many children won't realize this. Instead they'll stay immersed in discovery, reaching eagerly for each new idea, making their lessons more meaningful than with a "face-front" approach.

Young children learn best by experimentation and play. Kids are wired to grasp, drop, stack, and smash the world around them, often without adult encouragement. Problems start when students are taught things they can't see or touch. Math and grammar are difficult because they're less real than wooden blocks. Etoys makes abstractions more palpable, allowing children to visualize and explore new ideas.

Adult scientists utilize mathematics and literacy skills in the scientific process. Thus, it is important to weave mathematics and literacy knowledge into the development of science reasoning to allow this natural process to flourish in an integrated environment of thought. This allows the levels of mathematical, conceptual, and science reasoning to rise together. The long history of applying mathematics to the physical world goes as far back as Archimedes and geometry and is as recent as the application of symmetry in understanding elementary particles and cosmology. This integrated approach represents a spiral in understanding science through mathematics, building conceptual knowledge and expressing ideas through language. Science, mathematics and literacy can be woven together in the tapestry of the Etoys environment.

1.2 Features & Benefits

- Etoys is a free and Open Source multimedia computer environment.
- The Etoys community continually improves functionality, adds tools and fixes bugs through regular upgrades.
- Etoys runs on many different platforms including Sugar on the OLPC XO, Mac OS, Linux, and Windows.
- Etoys is used worldwide and is translated into many languages.
- Etoys is a learning environment that allows users to explore and create through authoring their own projects.
- Using Etoys makes learners' thinking visible.
- Using Etoys deepens understanding.
- Etoys is a interpreted language with extreme late binding, so users get immediate feedback when doing things like running scripts they create.
- The Squeakland web site has many example projects from users around the world and has links to many other active Etoys sites.

1.3 Etoys in the classroom

Etoys is a tool that can be used by teachers to:

- Create Curriculum
- Introduce new topics
- Assess children's understanding
- Provide skill practice
- Motivate students
- Help children learn **useful** and powerful ideas

Etoys is a tool that can be used by students to:

- Develop a deeper understanding
- Creatively explore new topics
- Create Reports

- Communicate Ideas
- Create their own Stories
- Learn from each other

GETTING STARTED

License: *The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, **you agree** that your edits can be used under both the **GPL and MIT** licenses.*

2. Getting Started

The best way to get started with Etoys is to try the interactive tutorials. These can be found in two places. They can be found on the Squeakland website at <http://www.squeakland.org/tutorials/demos/> or, they can also be found within the downloaded application software. Once you launch the Etoys application, click on the "Tutorials and Demos" cloud and the "Gallery of Projects" cloud on the welcome page.

These tutorials take you step by step through some of the basics of Etoys. Each one has an "About" tab that you can read to find out more about the topic introduced in the project.

Don't be intimidated by the depth of these tutorials and examples! Though the first ones take you through the very basics of defining "objects" and the "halo," some of the other tutorials are quite complex and are offered as an illustration of the range of activities that can be created with Etoys.

Another way to find out how complex the modeling can be is to view the Showcase at the Squeakland website. This is where Etoys users from around the world upload their work to share with the international community of Etoys users. Looking at the scripts can help you discover new capabilities.

Finally, the Help files within Etoys are excellent. Click on the question mark in the top left of your screen to see a list of help topics.

2.1 Installing

First, download the software. You can find it at the Squeakland website (<http://www.squeakland.org/download/>). Free downloads are available for Macintosh, Windows, and Linux operating systems. It is also available for the One Laptop Per Child XO computer.

There is a mobile version called Etoys-To-Go which you can download to your USB-Stick. You can put this stick on any computer and start Etoys without any installation directly from the stick. All your projects will automatically be saved on the stick. Since you have installation on the local computer, the browser plugin does not work if you use Etoys-To-Go.

Windows Installation

1. Click on the download link on the Etoys download page.
2. When the dialog box appears, Choose Save.
3. Save the file on your hard drive in a temporary folder. If the folder does not exist, you can create a folder by clicking on the folder with the asterisk in the menu bar.
4. Choose Start - My Computer - C: - Sugar - the install file (Etoys4-final-win).
5. A Dialog box will appear. Choose Run. When the License Agreement appears, click I Agree.
6. Choose a folder to install the program (i.e. C:\Program Files\Etoys)
7. When the Choose Start Menu Folder dialog box appears, click Install
8. Once the Install is completed, Click Close.
9. To run the program, choose, Start - A;; Programs -> Etoys -> Etoys
10. Enjoy!

MAC OS Installation

This installation was tested with Macintosh OS 10.5.8 and Mozilla Firefox.

1. Click on the download link on the Etoys download page.
2. When the dialog box appears, choose Save File.
3. Double click on the Etoys4-Final-Mac.zip icon. This will unzip the installer file.
4. Double click on the Etoys 4 Installer.
5. A Dialog box will appear. Read the information and click Continue. Follow the prompts and agree to the licensing terms.
6. Choose a location to install the program (i.e. Macintosh HD)
7. Click Install.
8. Once the Install is completed, click Close.
9. To run the program, choose Etoys 4 from within the Applications folder.
10. Enjoy!

XO/Sugar Installation

Etoys should come pre-installed in each XO machine. In the event that it's missing or deleted, here are the steps to take to get Etoys in this hardware/OS configuration:

.....

Etoys-To-Go

Etoys-To-Go is a version of Etoys that can run without installation directly from a USB drive. Projects are saved to the USB drive by default. All versions should run on Mac, Windows, and Linux. Follow the following steps to download and install Etoys-To-Go on our USB drive (Installation instructions based on Windows Vista):

- 1) Go to <http://squeakland.org/download/>
- 2) Click on *Etoys To Go*. This will open a dialog box in Windows Vista that will ask if whether you'd like to open the file Etoys-To-Go4-Final.zip (or latest version) or Save it in a directory. Click "Save File", which will put the file in the Downloads directory.
- 3) Open the Downloads directory. Double click on Etoys-To-Go4-Final.zip this will reveal the file Etoys-To-Go4.app(or latest version of software)
- 4) Copy Etoys-To-Go4.app (or latest version) to your USB drive.

The main difference between Etoys-To-Go and the regular Etoys is the way it finds the default project directory. In Etoys-To-Go, the projects directory is in the same place as the Etoys-To-Go application. Presumably that is a USB memory stick, so projects are stored on the stick, too, and can easily be transported. In regular Etoys, projects are saved to the user's documents directory on the machine's hard disk, so transporting them to a different machine needs copying.






Another difference is that the web browser plugin requires installation on the user's machine, because the browser needs to find the plugin, and the plugin needs to find the Etoys application. So Etoys-To-Go does not provide a web browser plugin.

Running the Etoys-To-Go Application

- 1) Insert the drive to a USB slot in your computer. Make note of the drive location on your computer.
- 2) In Windows, you can double-click the drive to reveal the contents of the drive. If it's a newly created drive, you'll see two folders: ".etoys" and "Etoys-To-Go4.app"(or latest version with .app at the end of the folder name).
- 3) Double-click the ".app" folder. Amongst the files contained in this folder is the Etoys application indicated by the Etoys logo.
- 4) Double-click the Etoys application to run/launch it. You will see the familiar welcome window.

Saving and Re-running Your Project Files using Etoys-To-Go

When you first save a project, an "Etoys" folder is created within the USB drive. This will contain all your project files (or PR files). Each subsequent saving of your project file will result with the original file name that you used plus an incremented number at the end. Example:

 PhilippineMap.001	11/16/2009 10:50 ...	PR File	102 KB
 PhilippineMap.002	11/16/2009 11:54 ...	PR File	102 KB
 PhilippineMap.003	11/17/2009 12:03 ...	PR File	104 KB
 PhilippineMap.004	11/17/2009 12:09 ...	PR File	110 KB
 PhilippineMap.005	11/17/2009 10:58 ...	PR File	126 KB

This is useful so you can always go back to old versions of your projects and work from there.

When you first go back to run your project, it will be tempting to double-click on the project file from the "Etoys" folder or directory. Unless Etoys is also installed in the machine, this will not run Etoys or open your project file. This is another difference with Etoys-To-Go and the machine installed version of the Etoys. With Etoys-To-Go, you have to launch/run the Etoys application first and THEN open the file from Etoys.

2.2 Loading, Saving, Quitting

Open Etoys and you will see a car driving around a screen, bouncing off the sides and off of three clouds. This is a project. This is also the main Etoys opening page. You can see the script that controls the car, along with suggestions for changing the variables in the script to see how the car responds.

The three clouds suggest places to start:

"Gallery of Projects" will take you to a selection of twenty two projects that you can look at and explore to learn some of the many things you can do with Etoys.

"Tutorials and Demos" will take you to three projects that will help you get started with learning Etoys.

"Make a Project" will bring you right into a blank Etoys project page, where you can create and manipulate your own objects.

Loading

To load a project into your Etoys application, click the find-icon.



If you hold the mouse button pressed for some seconds, there will be a menu with more options for loading projects. These options differ in the places where you will look for the project and in the kind of files you can open. The first option is the same that happens if you just click on the icon. It will open the default Etoys folder on your computer (or the folder on your usb drive if you are using Etoys-To-Go) to save your project. Here you can also choose "My Squeakland" and "Squeakland Showcase" to open a project directly from the squeakland website. Keep in mind that you need internet access to do this!

The second option lets you search throughout your whole directory structure.

The third option allows you to load images into your project. The currently supported formats are .jpg and .png.

Saving

Once you have a project, you'll want to save it so that you can work on it again and pick up where you left off. Saving a project is different on the XO than it is on a Macintosh or Windows computer.

Saving to an XO

If you click the X on the top right corner of the menu bar, your XO will automatically save to the Journal. You will know it's saving because you'll see a stick figure doing jumping jacks.

To retrieve a saved file from the XO, click on the folder with the arrow pointing up. Choose the project you want to work on and it will load into Etoys. (It's a good idea to name your projects so that when you go back to work on them, you can distinguish them from one another.)

Saving to a Macintosh or a Windows computer

Near the right corner of your menu bar, you'll see the "Publish (save)" icon.



When you are ready to save a project, click on the Publish icon, and you will see a screen waiting for you to "describe this project." Fill in a project name, description, your name, subject, target age and region, then click OK.

The next screen you see where you will be saving the project. You have two choices of places to save your project. If you save to "etoys" you are saving to your local computer. It's a good idea to save locally until you feel you are ready to share your project. If you save to "My squeakland" you will be saving to the Squeakland website, where you can choose to make your project public so that others can view it. You will need to create a Squeakland account if you want to save to My squeakland. Login, and click "Save" to upload your project to the Squeakland website. Don't worry, Squeakland doesn't sell or rent your information.

Whether you choose to save to "My squeakland" or to "etoys" you will be able to retrieve and edit your project to make changes.

To open an existing project from a Windows or Macintosh computer, click on the "Find (load) icon, login if you aren't already, and find the project you wish to work on.

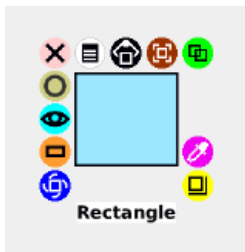
Quitting Altogether

To quit Etoys, simply click on the X on the top right of the menu bar. You will be prompted with "Are you sure you want to quit Etoys?" Select Yes, and your Etoys window will disappear.

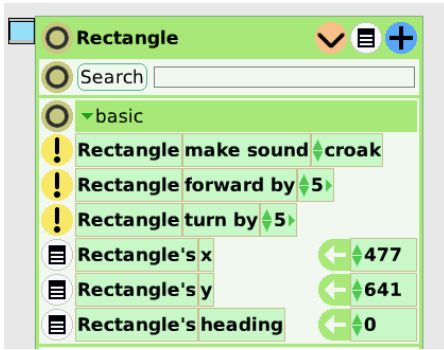
2.3 How to make a Script

Scripts are instructions written in a computer language that tell the computer what to do. In the case of Etoys the script is constructed by using graphical tiles that can be added together to create a set of instructions. The computer language behind the graphical tiles is Smalltalk, but you don't need to know the language to construct a script in Etoys. Every thing in the Etoys world is an object and you can use the tiles to tell any object what to do.

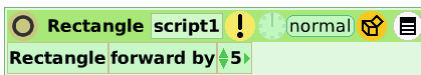
Start by opening Etoys and clicking on the "Make Project Cloud". You will see a gray blank page. That is the world or stage where you will have your objects or actors perform by telling them what to do with scripts. Click on the Supplies bin, the box at the top middle that looks like a toy box, to find objects to work with. Select a Rectangle object, click on it (it will stick to your cursor) and drag it out onto the world.



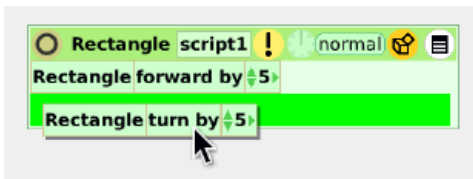
The next step is to open a Viewer for the object. Right click on your object (on a Mac command or apple click) and you will see a halo of actions in the form of small colored, circular buttons surrounding your object. Click on the light blue button and the center left of your object to open a viewer.



On the left you see a viewer for the Rectangle object. There are many categories of tiles that you can read about in the Common Tiles section. For now, we will focus on the basic category that you see to the left. The tiles are the 5 green rectangles that are below the word **basic**. The easiest way to create a script is to grab a tile and pull it out onto the world.



Here we have pulled the **Rectangle forward by** tile out onto the world and a ScriptEditor is automatically opened with the first instruction for the Rectangle object to go forward by 5 pixels. You can run the script once by clicking the yellow exclamation point at the top center or run it continuously by clicking the clock to the right of the exclamation point. There is much more information about the ScriptEditor in the User Interface section.



To add more commands to the script, simply drag out more tiles and add them to the script. The ScriptEditor will show green when it is ready to accept the tile. Simply let go of the tile and it will drop right into the script. Now when you run the script, the Rectangle object will go forward 5 pixels and turn 5 degrees.

This was just a start on scripting. You can read much more about using Etoys in the following chapters.

2.4 Object as Player

2.5 Showcase

The Squeakland Showcase is a place where you can see many examples of other people's projects. These projects range from student work to examples made by professional developers. It's a place where YOU can also share your work with the Squeakland community.

If you go to the Squeakland website, you can view the Showcase in a variety of ways.

"**Featured**" is a collection of projects selected by the Squeakland Education Committee for their illustrative qualities. They are chosen because they are good representative examples of the variety of things you can do with Etoys.

"**Everyone**" shows all the uploaded projects, which you can sort in a variety of ways:

- "**rank**" -- When a project is submitted, it gets rated and reviewed by some volunteers from the Squeakland community. Projects are ranked from 1-10, and the highest ranking projects show up first on the list.
- "**subject**" -- Subjects include Language Arts, Mathematics, Science, Music, and Visual Arts
- "**age**" -- Authors can specify "target age" for the project, which include from ages 4 and older, ages 8 and older, ages 11 and older, and ages 15 and older.
- "**region**" -- Authors can specify where the project is from, which can sometimes help you find projects in your native language.
- "**tag**" -- Authors can specify tag words to help you find projects. These might include something like "fractions" to distinguish this project from other mathematics projects.
- "**date**" -- Sorts the Showcase by date, with most recently posted first.

"**Signup**" brings you to a page where you can create a Squeakland account and join the Squeakland community. Don't worry, we won't sell, rent, or otherwise give out your information, though we do encourage you to sign up for the newsletter and other occasional updates.

"**Upload**" brings you to a page where you can upload a project, though it's easiest to do this right from within Etoys. See Instructions in the chapter on Saving a project.

***License:** The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, **you agree** that your edits can be used under both the **GPL and MIT** licenses.*

3. User Interface

3.1 Overview

3.1.1 Toolbar

The toolbar at the top of the screen includes basic navigation through Etoys. Each of these has a balloon help bubble that pops up if you hover over any item. Following is a list of each of the toolbar items.



In **Etoys 4.1** the toolbar has changed, the icon for the new project has been removed.



Here comes a list what the individual icons mean.



When you click here, it opens the Quick Guides, an interactive Etoys book of help topics. In **Etoys 4.1**, there are other languages than English available. The Quick Guides will be shown in the chosen language, if available, otherwise in English.



This shows the current name of your project. You can change the name directly by writing in the textfield, or you can save the project and give it a name during the process.



This creates a new project within the current project. This can cause problems when it has been clicked while another project than the home project is currently shown. **This button has been removed in Etoys 4.1.** To create a new project, go to the home project and click on the cloud with the text "New Project".



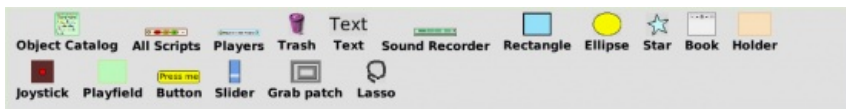
Click on the arrows to go to either the previous or the next project in your list. The left button brings you back to the project which you just left (the previous project). The right button brings you to the next project, if there is any.



Here you can open the painting tool and create sketches. Every time you click on the icon, you will open a new sheet and start a new painting. Make sure to quit the painting tool when you finish drawing. To redraw an existing painting, use the gray halo icon.



A click on this icon opens a flap, which is called supplies bin. Here you can find a great library of objects ready to use.



A user can customize the objects that are show in the supplies flap. Almost all objects from Object Catalog can be added to the Supplies flap except for *Particles* because it consists of several parts (Kedama World, patch, turtle). To do this, drag the object you want first to the world, then open the supplies flap and drag it to the flap. Dragging the object from the Object Catalog directly to the Supplies flap will not work. Note that the changes to the supplies bin will only last until you quit Etoys. It will not be saved with the project.

You can remove any of the objects in the Supplies Flap easily using the pink icon from the objects halo. Removing the object from the Supplies Flap doesn't delete the object permanently but just deletes the object's icon.



Here you can change the language. When you start Etoys, the language you have chosen on your computer will automatically be used.



When you click once, it toggles the full screen view. When you hold down the mouse button, you can also switch off scaling. The default setting is optimized for the resolution on the XO laptop.



You can open a saved project from your default Etoys folder when you click on this icon. When you hold down the mouse button, you can search in other folders or open other files.



You can save a project to your default Etoys folder by clicking on this icon. Hold down the mouse button to get more options. You can save the project to other places or publish it directly to the showcase.



Click here to quit Etoys.



Click to hide or show the toolbar. This is helpful for projects made in older Etoys versions (before 3.0), so that the toolbar will not hide parts of the project.

3.1.2 Toolbar (Sugar)

On Sugar, the toolbar looks slightly different.



In **Etoys 4.1**, the stop-button has been replaced by an exit-button.



Collaborate with other Etoys users in the network. To share your Etoys activity you click the Share button and select "My Neighborhood". This makes the activity public and its icon will be visible in everyone's neighborhood view. To join that activity, a buddy would click that icon in the neighborhood.

Alternatively, you can invite a buddy to your Etoys activity. This will not make the shared activity visible to everyone, but just to the invited buddies. To invite someone, zoom out from Etoys to the Sugar neighborhood view. From the menu palette of the buddy's icon select "Invite". They will get a notification and can join your activity by clicking it.

In either case, when a buddy joins your activity, a new flap labeled "buddies" will be created on the left screen edge. A "badge" representing the buddy will be placed there. Similarly, on the joining buddy's screen a similar flap will be created. The buddy's project will be otherwise empty, your project is not transferred to them. When more buddies join, everyone will get badges for everyone else.

The badge is the way to communicate with the joined buddies. You can send an object to a buddy by dragging and dropping it onto the buddy's badge. The object with all its behavior and scripts will be copied to the buddy's computer. As soon as the transfer is finished, a signal tone is played and the object is attached to the buddy's pointer.

If you click the small "C" button on a buddy's badge, a text chat with the buddy is opened. You can use this to talk to each other. To stop collaborating, click the Share button again and select "Private". Or simply stop Etoys.



Click here to find an entry in the sugar journal and insert pictures etc. Hold mouse button down to see more options



Click here to keep a copy of your current project in the sugar journal. It will be stored in a new Journal entry (independent of the entry that is created or updated on stop ping). This statement is no longer true for **Etoys 4.1**. You will need to hit the "Save" folder to store a new project entry in the Journal. Hold mouse button down to see more storing options, like saving the current project to local storage or to a server.



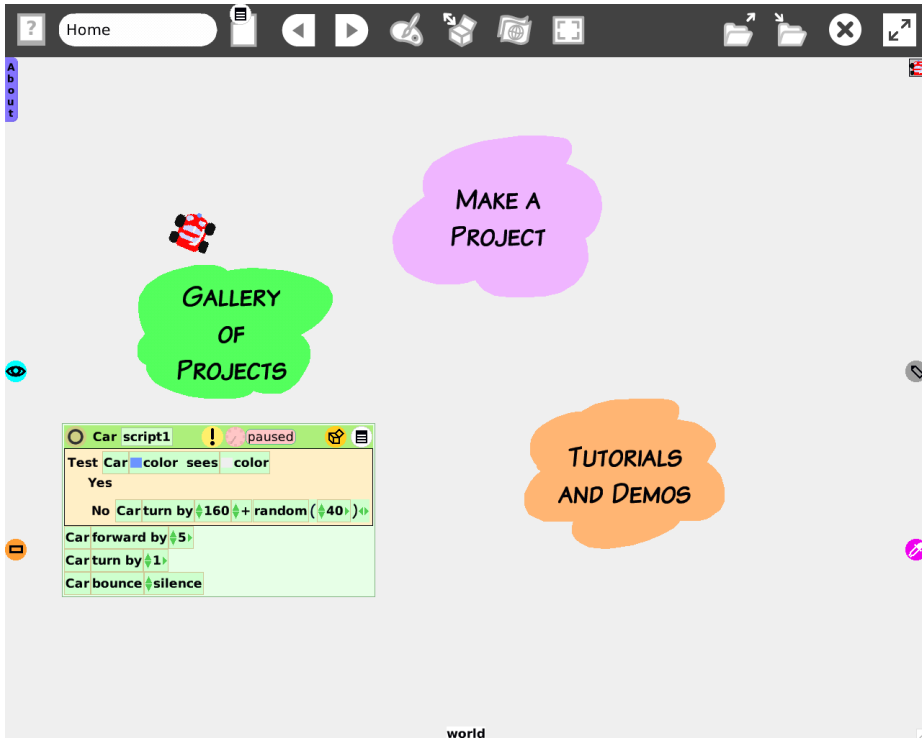
Click here to exit Etoys. This results to an automatic save of the project in the Journal. The home project is not stored on exit unless you rename it in the tool bar. This is to prevent useless copies of the home screen in the Journal. If the current project was loaded from the Journal, that Journal entry will be overwritten. Otherwise, a new Journal entry is created.



In **Etoys 4.1**, this is the button to exit Etoys. Exiting Etoys no longer automatically saves the project in the Journal on version 4.1. Upon hitting the "X" to exit, the user will be reminded to save the project before exiting.

3.1.3 World

(See further information about the World in Chapter 5 of this manual.)



The *world* is the main project page you see when you open an Etoys project. It is basically a playfield (see chapter 5, playfield). Because of that the world can contain several individual objects within it. The home screen shows

- an about-flap with the information of your Etoys version,
- an open script, a running car, and a folded viewer for the car,
- three clouds to go to the gallery of projects, tutorials and demos or to create a new project.

Tutorials and Demos

In this section you'll see three demos, of increasing difficulty. We recommend that if you are new to Etoys, you start here. Go through the tutorials so you can learn about how to navigate the Etoys world and get the most out of your time.

Gallery of Projects

If you want to see examples of what you can do with etoys, look at the various models shown here. For further exploration, go to <http://squeakland.org> and view the online Show case for even more projects and project ideas.

Make a Project

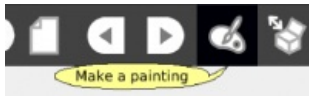
When you click on this cloud, you will see a blank screen. At the top is your menu. You'll even see two call out bubbles giving you two places to start. If you hover over any item you will see a call out bubble describing what that item does.

To change the background of the world, use the gray halo icon. The painting will be automatically embedded into the world and inert to all user interactions by default. To change this behavior, use the menu icon of the halo.

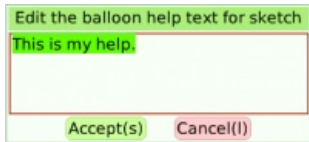
When you first open Etoys, you'll see a car driving around a screen. This is itself a model. The car tests if it is bumping into things, and if it does, it bounces off and drives off in a different direction.


3.1.4 Balloon Help

Hovering over an item in the toolbar will bring up balloon help telling you what each item does.



Balloon help is Etoys text, users can edit it. To add a balloon help to your own sketch, bring up the halo and click on the menu icon. From the menu entries, choose "extras" and then "balloon help for this object". Then you can type or change the balloon help text.



The help will be shown when you hover over your object or when you click the question mark.  Balloon help might be available in your local language. If it is not, and you want to help with the translations, please read in the Appendix how to do this.

3.1.5 Flaps

Flaps are a place to store objects for later use or put in notes. Special flaps can be used to share objects between projects. To get a new flap, there are some options:

1. Use Cmd-, and a menu will pop open with an option "flaps..."
2. Cmd-Shift-W offers the full "dangerous" Squeak menu, dangerous because it allows to save the image which is not a good idea unless you know what you're doing.
3. Make a copy of the flap from the home page and drag it into the supplies bin. Then you can drag it from the supplies bin and put it in another project. You will need to delete all the information in the flap, but this avoids exposing the other items from CMD-, menu and shows how to use the supplies bin to copy objects to other projects.

In **Etoys 4.1**, a flap to add to your projects is already in the supplies bin.

3.2 Projects

Projects are the unit you work with in Etoys, what you are saving and loading. When clicking on the "Make a Project" - cloud, the homescreen disappears and you get a whole empty world. Everything you put in the world belongs to the project and will be saved. You can not add another project to the project you are working on! Only the project you are currently in will be saved. So if you have nested projects, the hierarchy and all other projects will not be saved.

If you want to show several projects from within Etoys in a certain order, use the project navigator (see chapter 5).

3.3 The Halo

Every object has a "halo" to allow you to modify it in various ways. Access the halo of any object by right clicking on the object. The tip of the arrow is the "hot spot." Clicking on a very thin line can be a bit tricky because you have to aim the arrow correctly.



This is the halo you get for a sketch. There are other objects with aslightly different set of icons.

-  The pink "X" at the top left will delete the object.
-  The tan "O" along the left edge will collapse the object back into the viewer.
-  The blue eye in the middle left will open a viewer for the object.
-  The brown circle with the rectangle will make a "tile" for the object.
-  The blue spiral at the bottom left corner will rotate the object.
-  The yellow square in the bottom right will resize the object if you click on it and drag. If you hold down the shift key while resizing the aspect ratio will be maintained. Note: If the heading is changed and does NOT equal zero this icon will only let you change its scale.
-  The gray pencil will allow you to redraw the object, if the object is a sketch. (Useful if you wish to duplicate the object, then change the drawing slightly). If the object is a play field it will allow you to draw an background image on the playfield.
-  The green double boxes in the top right duplicates the object. If you hold down the shift key you will create a sibling object. There is a difference between siblings and dupli cates. See more about this in the chapter AppendixMorph.
-  The red-brown shape to the left of the green duplicate button allows you to move the object.
-  The black button that looks like a house lets you pick up an object and place it some where else.
-  The shape that looks like a newspaper in the top row gives you a menu of other options. What options you find here exactly depends on what kind of object you are look ing at. It will be explained in more detail with each object in chapter 5.



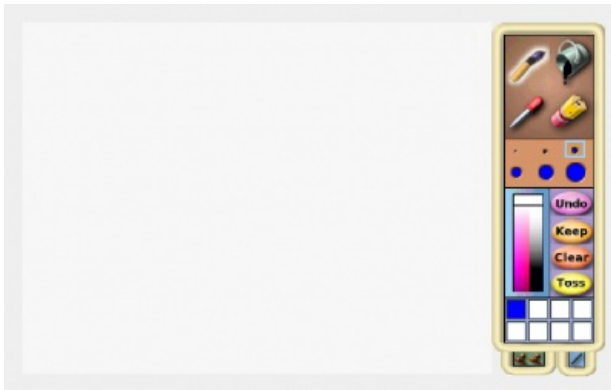
The center handle with the arrow on it changes both the direction of "forward" (hold down the shift key and drag the tip of the arrow to the direction you want the object to move) and the center of rotation (hold down the shift key and drag the center of the circle to the point you'd like the object to rotate around - when you use the turn command.)

Different types of objects will have different halos. For example, a text object will have halo handles to change the font size and style.

3.4 Painting

When you click on the paint palette in the toolbar, the painting tool will appear and part of the screen is covered with a transparent rectangle. This is your drawing sheet. It is smaller than your screen. Most of the times, you want to draw objects to move them around later using scripts, so your drawings shouldn't be too big. Everything you draw on one page will be one object. If you want to draw several objects, exit the painting tool by clicking "keep" when one object is done and open it again to get a new sheet to draw on.

If you want to draw the background of the world, use the world's gray halo icon.



The paintbrush uses the brush size selected in the six circles beneath, and the color selected in the colorchart.



The bucket pours the selected color on the canvas and fills the whole region of the color under the tip of the pour. If a region is not completely surrounded by a color the fill will spill through and fill a huge area. Be careful!



The colorpicker picks up the color under its tip. You can use it over any object to get its color. A big color picker chart pops up when you move the mouse over it.



In the top row you can pick up *no color* or transparent to take away all color with the selected tool. Click and hold mouse down to move around, paint brushes indicate which color is being picked out.



The Eraser erases the the colors to transparent.



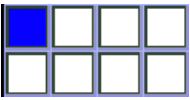
Here can you choose the size of your brush or eraser. The brush uses a round shape and the sizes are in pixels 3, 7, 13, 25, 50, 80. The eraser uses a square brush the width of the selected paint brush.



Clicking on **Undo**, you can undo the last change you made. It remembers one change. Click again to redo the change.

Click on **Keep** when you are done editing the drawing to exit the painting editor. When you click on **Clear**, your whole sketch will be erased and you have a blank sheet again.

Toss throws away the current changes and exits the paint editor. It brings up a dialog asking if you really want to throw away your painting. If you made a new painting, throw away will erase everything. If you were editing an existing painting, throw away will give you back the painting like it was before you started to edit it.



Color Swatches keeps the 8 recent used colors for fast access.



Click on the flap below the painting box to open the shape tools. This opens a selection of shapes you can select.



Draws a line straight line. Click mouse and hold down, move mouse and release to make line. It uses a square brush with current brush size and selected color. Hold the shift key to step lines angle in 45 degree intervals



Draws a rectangle in current color and brush width. Hold the shift key to make a square.



Draws an ellipse in current color and brush width. Hold the shift key to make a circle. Ellipse draws from center.



Hands you a polygon in current color and brush width. Move the yellow handles to change polygon. Drop yellow/blue on top of each other to erase a handle. Click and move green triangles to make more handles. Click outside handles to end editing.



Draws a 5 pointed star in current color and brush width. Draws from center, click and drag to select size.



Click on the left flap below the paint box to open the stamping tool.




The stamp is used for making a copy of a selected part and stamping it back to make patterns or drawings. Click on a stamp, then select the area you want to copy of your drawing with the mouse. Your selection will be shown in the blue rectangle below the selected stamp. You can now stamp your selection. If the picture you are drawing does not have a background fill color, the stamp tool will make the stamp without a background color.



3.5 Viewers

A viewer holds the tiles that describe all the attributes of an object. Use it to script your object to define how it behaves. Bring up the halo for your object and click on the blue eye in

the middle left  to open a viewer for this object. A viewer is always representing the state of the object it belongs. You can open several viewers to get access to the tiles of different objects. You can use tiles from different viewers in your script.



In the top row of the viewer you find the name of the object you are viewing. To change the object's name just click on it, and you can edit it.

All viewers have a little thumbnail flap, much like a tab on a folder, so its easier to identify which object you are viewing.

You will find specifics about the tiles you'll see in the viewer in the "tiles" section of this manual, but in general, the viewer is organized by category. There are approximately fifteen different categories in a viewer, but the number changes depending on what object the viewer belongs to. For example, a "Holder" has some categories that a "Sketch" does not have and vice versa.

The categories are separated by green stripes. The title of the category is written inside the stripe. When you first open a viewer, you will see the "Basic" and "Tests" categories. Click on one of the green stripes to change the category.

There are 4 different types of tiles: some have an exclamation mark on the beginning, some a small menu icon, some have no icon at all and some have a light orange background color.

The tiles with the yellow exclamation mark are commands. They are executable. Click the exclamation mark once and the object will execute that command (e.g. forward 5) one time. If you drag one of these tiles onto the world, Etoys will automatically generate a script.

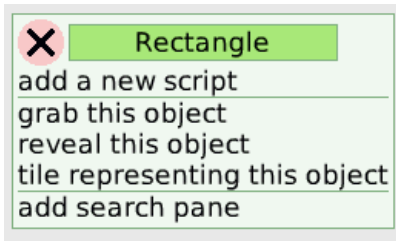
Tiles with the black and white menu icon next to them are attributes of the object, which have an assigned value. You can access either the value of the object by dragging the tile by it's name or an assignment command by dragging it on the green assignment arrow. By clicking on the menu icon, you can open a watcher for the values and change, how many decimal places will be shown for numeric values.

A watcher will show the current value of the attribute (together with the name in a detailed watcher) and give the user the opportunity to manipulate it. So you don't need to open the whole viewer window to see certain values.

Some tiles in the test category have no mark next to them. These tiles are conditions which are to be used with the test tile.

The viewer menu

You can open the viewer menu when you click on the menu-icon in the top-row of a viewer. It is in the middle between the variable icon and the category pane icon.



The viewer menu items do the following:

- add a new script: Attaches a new script tile to the mouse that expands to a ScriptEditor when dropped.
- grab this object: Attaches the object to the mouse.
- reveal this object: Make the object flash a few times so you can easily see it on the screen. This command also brings the object back on screen if it has moved beyond its borders somehow. If you used the command Object hide, 'reveal this object' will flash it, then bring up the halo.
- tile representing this object: Puts a tile representing this object in the mouse
- add search pane: Adds a search pane to the viewer.

Categories

Tiles are divided into categories that access or share topic. You can add another category pane when you click on the "+" sign in the blue circle right beside the menu icon.

All objects share these categories:

- Variables
- Scripts
- Basic
- Color
- Geometry
- More Geometry
- Pen use
- Test
- Motion
- Fill & Border
- Scripting
- Sound
- Observation
- Drag & Drop
- Miscellaneous

Some objects have specific categories that give access to the object's abilities. For example, "Button" also has a button category that allows you to change the look and use of the button. "Star" has a star category that allows you to change the number of points on the star, and objects like "Playfield" and "Holder" have categories specific to them.

Variables

Variables keep information stored within your program. They can contain different types of information and you set the kind of information the variable stores in its menu. Variables should be named to indicate their purpose. The name of the variable helps you when you or others read your script later on. You can add a variable by clicking on the "v" sign in the icon left of the menu icon.

- Boolean
A Type with two possible values: true or false
- BorderStyle
A Type with the following possible values: simple, raised, inset, complex framed, complex raised, complex inset, complex alt framed, complex alt raised, and complex alt inset. This variable could be used in a script to create the visual effect of a Button being pushed.
- ButtonPhase
A Type with the following possible values: button down, while pressed, button up
- Color
A Type that stores a particular color. A Color is made up of multiple attributes as shown in the color category of the viewer. When you set a color all those values are updated. A color variable can be used in tests, to change the color objects in your world. See the Common Tiles chapter for a description of the color variables.
Etoys Challenge: Take an existing project and use a set of color variables to allow you to change color schemes for your project.

- **Graphic**
A Type used to store a graphic.
- **ImageResolution**
A Type used by the Camera object (found in the Multimedia section of the Object Catalog). This can be used to change the resolution of the image captured. Possible values are: original, 256 colors, 256 grays, 4 grays and black and white. Lower resolutions will reduce the size of the object.
- **Number**
A Type used to store a number. The number defaults to displaying 1 decimal places to the right of the decimal point. You can specify the number of decimal places that will be displayed. NOTE: while you may only display 1 decimal place to the right of the decimal point, if its actual value is 2.45, it will display 2.5. But, if you do a Test for myNum >= 2.5, the test result will be false and the NO branch will be executed.
Pop Quiz: Can you determine under what cases the display of a number will round up?
Pop Quiz 2: Can you determine how many decimal places can be accurately represented in an Etoys Number?
Pop Quiz 3: What is the largest number you can use in Etoys?
- **Patch**
This variable Type currently does not work in Etoys 4 and will throw an exception "Message Not Understood". For now do not use this variable type.
- **Player**
This Type is used to store a reference to an Object. In Etoys the terms Player and Object basically mean the same thing.
- **Point**
This Type is used to store an X,Y location. You must specify it in the following format X@Y where both X and Y are numbers (not variable names) so: 100@200 is valid and specifies X=100, Y=200. While you can enter a single digit, if you try to use that Variable in a script to set an Object's location an Exception message will appear.
- **ScriptName**
This Type is used to store script names. A list of existing script names plus "empty script" will appear in the list. This can be used to scripting, where one script sets a script name variable, and another script uses that variable to determine which script to run. Script names with Capital letters will show in lower case with a space preceding the capital letter. So the Script "haveFun" will show as "have fun" in the pull down list.
- **Sound**
A Type used to store a sound object. The list will include all the default sounds, plus any sounds added to the project.
- **String**
A Type used to store a set of characters.
- **TrailStyle**
A Type used to store a pen trail style, valid values are: lines, arrows, arrowheads, and dots.

Search

Here you can search for a specific tile. This search is within the categories belonging to the current viewer.

3.6 Script Editor



Although some object actions can be controlled in the object's viewer, the script editor is where you assemble the tiles to create more complex actions or scripts. In the script above two of the object sketch's tiles have been dragged from sketch's viewer and dropped into the Script Editor. When the tile is in place to be accepted by the Script Editor, the area around the tile turns green and when the tile is released an audible click is heard if sound is enabled.

The Script Editor allows you to assemble tiles and test out your ideas for creating a simulation in the Etoys world. Getting into the habit of testing your ideas is a very powerful concept. You can use the results of these test scripts to simulate and determine the attributes and behaviors of the objects in your world. Getting into the habit of testing your ideas or at least thinking about how you could test your ideas (and what other people tell you is true) is a wonderful "habit of the mind". Let your imagination reign.

Collapse button

The round brown button with the black enclosed circle at the upper left collapses the script back to the viewer. The script can always be viewed in the Script Editor again by dragging it out onto the world from the object's viewer.

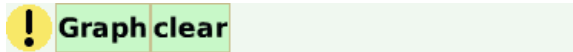
Name of the Player being Scripted.

The text to the right of the collapse button is the name of the object for which you are creating the script. You can not change the objects name here.

Name of script.

The text to the right of the object's name is the script's name. It's a good idea to give a good name to the script (e.g. If its a script that moves the player, you can call the script 'move', etc.).

Exclamation point.



The yellow exclamation point in the center of the Script Editor runs the script one time. You can also run the script from the object's viewer as shown above.

Tickindicator

The TickIndicator is the clock face to the right of the yellow exclamation point. The TickIndicator is light green when not in use, pink when script is paused and blue when script is ticking. Hold mouse button down and a menu pops up. There you can set the rate at which the script should run. The default is 8 ticks a second. Predefined in the menu are choices from one tick to 100 ticks a second. Last menu option is called *other...* Here you can type in the number of ticks per second that you want the script to run. A tick rate of 0.5 indicates one tick every other second, 0.1 once every 10 seconds, etc.

ScriptStatusControl

This appears to the right of the clock. Click it to get the following menu of options:

- normal - run when called
- paused - ready to run all the time, will change to "ticking" Status when you press the "Go" button in the "All Scripts" object.
- ticking - run all the time, will change to "paused" Status when you press "Stop" button in the "All Scripts" object
- mouseDown - run once when mouse goes down the object
- mouseStillDown - run while mouse still down on the object
- mouseUp - run once when mouse comes back up off the object
- mouseEnter - run once when mouse enters the object's bounds with the button up
- mouseLeave - run once when mouse exits the object's bounds with the button up

- `mouseenterDragging` - run once when mouse enters the object's bounds while dragging another object
- `mouseleaveDragging` - run once when mouse exits the object's bounds while dragging another object
- `opening` - run once when the object is being opened, this only works for World and Pages (contained in a book) objects.
Pop Quiz: Can you figure out how to use this to ensure a Book always opens to the first page when a project is opened? Can you figure out how to use this to reset a Page each time it is opened?
- `closing` - run once when the object is being closed. This only applies to World and Pages objects.

More events:

- `connectedTo` - run once immediately after a connector has connected to the object
- `disconnectedFrom` - run once immediately before a connector is going to disconnect from the object
- `KeyStroke` - executed when any key is pressed, this is only available from World and Playfield scripts
- `acceptedTextContents` - executed when text is accepted, which can be when you tab to the next text field or hit return (if *accept on CR* is checked), this is only available for *Enhanced Text* objects.

Gold chest.

Clicking on the gold chest to the right of the ScriptStatusControl allows access to the following special tiles:

test tile

This tile has three holders into which you can place one or more tiles:



Test: You can place any tile that contains an attribute of any object. For example, you could test whether Ellipse's y value is less than 5 as shown. To accomplish this grab the left side of the Ellipse's y tile from Ellipse's viewer. Make sure you aren't to the right of the tile and have the red box surrounding Ellipse's y and its value at the right, because then you will obtain an assignment phrase for the tile and not its attribute. You can test on almost any attribute (numeric, color, graphic, etc) of any object.

Pop Quiz: How could I define a test to see if a car passed a finish line in a race?

Yes: Tiles placed in the "Yes" holder will be executed when the "Test" is true. If $y < 5$, then Ellipse will turn blue.

No: Tiles placed in the "No" holder will be executed when the "Test" is false. If $y >= 5$, then Ellipse will turn blue.

Note, that many tiles can be used in the "Test" holder of the "Test Yes No" tile. Some can also be used as watchers, which can be useful when debugging your scripts. While you may think a "Sketch obtrudes" or a "Sketch is under mouse" the Etoys system may think differently. Being able to see this as a script runs can help.

repeat tile

This tile has one holder called "do" into which you can place a script:



You can repeat the script you drop into the "do" placeholder the number of times set in the number box at the top of the tile.

random tile



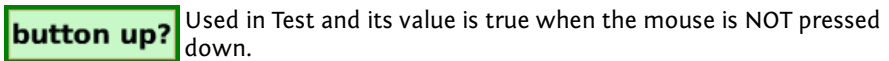
The random number will range from 0 to the number in the number box (in this case 5). It can be dropped into the end of a command that requires a number or function. This is a special case of a function tile.

function tile



You can choose a function by clicking on the arrows at the left or clicking on the function name (in this case "abs") to get a drop down list of functions to choose from. Most of the common functions are available in the list. The function can be dropped into the end of a command that requires a number or function.

button up?



Used in Test and its value is true when the mouse is NOT pressed down.

button down?



Used in Test and its value is true when the mouse is pressed down anywhere.

tile representing the player



In this case the player is Ellipse.

number



A number box that can be dropped into the end of a command that requires a number or function.

Script editor menu

The Script Editor Menu button appears on the far right. It has the following options.

add parameter

Many tiles in Etoys have a parameter. It is a value that influences the behavior of that tile.

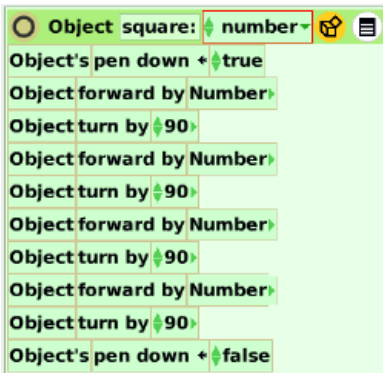
For example, the "forward by" tile has a parameter that specifies how far the object should move. The "make sound" parameter specifies which sound to make. Etc.

You can also make your own tiles. When you create a new script, a tile for it is placed in an object's "scripts" viewer category. That tile can be used in another script.

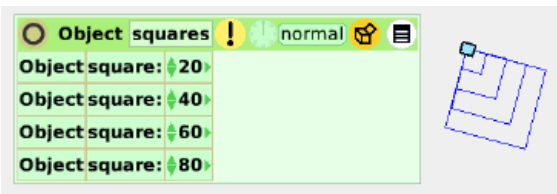
The "add parameter" menu entry lets you add a parameter to your own tile. If you use that new tile in another script, you can set a specific value for the parameter just like with any other tile.

Inside the script for your tile, that parameter is shown in the title bar. It is called "number" (unless you change the parameter type). You drag this "number" tile and use it like any variable tile in that script.

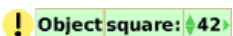
E.g. in this script, I used the "number" tile four times:



Every time you use your new tile, you can give it another "concrete" value. That's why in the script itself, you only have the "abstract" value called "number". (Computer scientists call the concrete values "arguments" and the abstract values "parameters", but even programmers get confused about the distinction).



Note that after adding a parameter, you can not set your script to "ticking" anymore, because it now needs a parameter. It only gets a concrete value for the parameter from another script where your tile is used. Additionally, you can set a value in the viewer, and hit the yellow button to run your script with that value:



The following are the data types that can be used as parameters:

- Boolean (true or false)
- BorderStyle
- ButtonPhase
- Color
- Graphic
- ImageResolution
- Number
- Patch
- Point
- ScriptName
- Sound
- String
- TrailStyle

button to fire this script

This will create a button object that can be used to fire the script. You can change the button's label from the menu in its viewer by selecting change label.

show arrows

When selected the arrows inside scripting tiles will be displayed (ex: the up and down arrows to increase numbers and the left and right arrows to remove and parts of the expression)

edit balloon help for this script

This edits the balloon help that is shown when you hover over the script name in the scripts category of the Objects Viewer

explain status alternatives

Opens a window with descriptions of the options for "when this script should run" (ex: normal et al)

show code textually

Will provide a text version of the script which you can edit. This provides an entry into Squeak programming. To save your changes type <ctrl><s> (on PC and XO) <cmd><s> (on Macintosh) when done. Note once changes are made you can not revert back to tile scripting without losing your changes.

save this version

When editing the text version of a script you can use this menu option to save the changes.

grab this object

Will move the object to the position of the mouse and you can drag it to where you want to place it.

reveal this object

Will display the halo for this object so you can find it. This can take a second or two.

tile representing this object

Will create a tile representing the script's Object. Tiles representing an object can be used in this or other scripts to replace the Object being acted upon in a script.

open viewer

Will open the viewer for this script's object

destroy this script

Will destroy this script.

3.7 Mouse Buttons

In Etoys, right-clicking the mouse will open the halo (the menu of colored circles) for the currently selected object. This will be the world, when you right-click after starting Etoys. When you keep right-clicking, you will select and open the halo for every embedded object in the structured you are pointing at.

Left-clicking will invoke a behaviour if you click on a button, menu entry or menu icon. When you left-click on an object, you can often move it around.

For any object, you can set it's behaviour regarding mouse-clicks. The menu icon in the object's halo provides the options "resist being picked up" or "be locked". The first one says whether the objects is resistant to a drag done by mousing down. The second option prevents all user interactions.

3.8 Keyboard Shortcuts

Many of the standard keyboard shortcuts work in Etoys.

Description	Windows	Macintosh	XO
Copy	ctrl-c	cmd-c	
Paste	ctrl-v	cmd-c	
Open System Browser window	ctrl-b	cmd-b	
Find (Works within a text box	ctrl-f	cmd-f	
	ctrl-i		
Choose font and size within a text box.	ctrl-k	cmd-k	
Redraw Screen	ctrl-r	cmd-r	
Open Transcript window	ctrl-t	cmd-t	
Choose justification options within a text box	ctrl-u	cmd-u	
Cut	ctrl-x	cmd-x	
Undo	ctrl-z	cmd-z	
Display/Hide Shared Tabs (fyi: the Sugar Navigator Flap is a shared tab)s	ctrl-F	cmd-F	
Display Object Catalog	ctrl-o	cmd-o	
Open World Menu	ctrl-W	cmd-W	
Open find a file list	ctrl-L	cmd-L	

FYI, I found the following in Etoys by cmd-W to open the "World" menu, then click on "help..." which will open a "Help" menu, then click on "command-key help) We should test these and update the page

Lower-case command keys

(use with Cmd key on Mac and Alt key on other platforms)

- a Select all
- b Browse it (selection is a class name or cursor is over a class-list or message-list)
- c Copy selection
- d Do it (selection is a valid expression)
- e Exchange selection with prior selection
- f Find
- g Find again
- h Set selection as search string for find again
- i Inspect it (selection is a valid expression, or selection is over an inspect-ilst)
- j Again once (do the last text-related operation again)
- k Set font
- l Cancel
- m Implementors of it (selection is a message selector or cursor is over a class-list or message-list)
- n Senders of it (selection is a message selector or cursor is over a class-list or message-list)
- o Spawn current method
- p Print it (selection is a valid expression)
- q Query symbol (toggle all possible completion for a given prefix)
- r Recognizer
- s Save (i.e. accept)
- t Finds a Transcript (when cursor is over the desktop)
- u Toggle alignment
- v Paste
- w Delete preceding word (over text); Close-window (over morphic desktop)
- x Cut selection
- y Swap characters
- z Undo

Note: for Do it, Senders of it, etc., a null selection will be expanded to a word or to the current line in an attempt to do what you want. Also note that Senders/Implementors of it will find the outermost keyword selector in a large selection, as when you have selected a bracketed expression or an entire line. Finally note that the same cmd-m and cmd-n (and cmd-v for versions) work in the message pane of most browsers.

Upper-case command keys

(use with Shift-Cmd, or Ctrl on Mac

or Shift-Alt on other platforms; sometimes Ctrl works too)

- A Advance argument
- B Browse it in this same browser (in System browsers only)
- C Compare argument to clipboard
- D Duplicate
- E Method strings containing it
- F Insert 'iffalse:'
- G fileIn from it (a file name)
- H cursor TopHome:
- I Inspect via Object Explorer
- J Again many (apply the previous text command repeatedly until the end of the text)
- K Set style
- L Outdent (move selection one tab-stop left)
- M Select current type-in
- N References to it (selection is a class name, or cursor is over a class-list or message-list)
- O Open single-message browser (in message lists)
- P Make project link
- R Indent (move selection one tab-stop right)
- S Search
- T Insert 'iftrue:'
- U Convert linefeeds to carriage returns in selection
- V Paste author's initials
- W Selectors containing it (in text); show-world-menu (when issued with cursor over desktop)

X Force selection to lowercase
Y Force selection to uppercase
Z Capitalize all words in selection

Other special keys

Backspace Backward delete character
Del Forward delete character
Shift-Bksp Backward delete word
Shift-Del Forward delete word
Esc Pop up the Desktop Menu
\
Send top window to back

Cursor keys

left, right,
up, down Move cursor left, right, up or down
Ctrl-left Move cursor left one word
Ctrl-right Move cursor right one word
Home Move cursor to begin of line or begin of text
End Move cursor to end of line or end of text
PgUp, Ctrl-up Move cursor up one page
PgDown, Ctrl-Dn Move cursor down one page

Note all these keys can be used together with Shift to define or enlarge the selection. You cannot however shrink that selection again, as in some other systems.

Other Cmd-key combinations (not available on all platforms)

Return Insert return followed by as many tabs as the previous line
(with a further adjustment for additional brackets in that line)
Space Select the current word as with double clicking

Enclose the selection in a kind of bracket. Each is a toggle.
(not available on all platforms)

Ctrl-(Enclose within (and), or remove enclosing (and)
Ctrl-[Enclose within [and], or remove enclosing [and]
Ctrl-{ Enclose within { and }, or remove enclosing { and }
Ctrl-< Enclose within < and >, or remove enclosing < and >
Ctrl-' Enclose within ' and ', or remove enclosing ' and '
Ctrl-" Enclose within " and ", or remove enclosing " and "

Note also that you can double-click just inside any of the above delimiters, or at the beginning or end of a line, to select the text enclosed.

Text Emphasis

(not available on all platforms)

Cmd-1 10 point font
Cmd-2 12 point font
Cmd-3 18 point font
Cmd-4 24 point font
Cmd-5 36 point font
Cmd-6 color, action-on-click, link to class comment, link to method, url
Brings up a menu. To remove these properties, select more than the active part and then use command-0.
Cmd-7 bold
Cmd-8 italic
Cmd-9 narrow (same as negative kern)
Cmd-0 plain text (resets all emphasis)
Cmd-- underlined (toggles it)
Cmd-= struck out (toggles it)

Shift-Cmd-- (aka _) negative kern (letters 1 pixel closer)

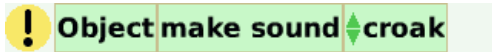
Shift-Cmd++ positive kern (letters 1 pixel larger spread)

License: *The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, **you agree** that your edits can be used under both the **GPL and MIT** licenses.*

4. Common Tiles and Menu Items

Tiles are the blocks from which you can build Etoys scripts. You will find the tiles you can read about in this chapter in most of the objects within Etoys. The chapter is sorted by categories. There are special categories for special objects and you will find their explanation in the chapters where the objects are covered. Sometimes you will find fewer tiles in a category or fewer categories than are discussed in this chapter. For example, the category color doesn't exist for all objects. It should be noted that the location of an object is the location of its rotation center. The main headings below are the categories and the headings beneath them are the specific tiles in that category.

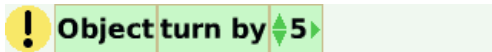
4.1 Category Basic



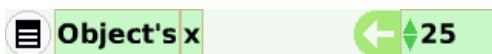
The object will make the sound which is specified in the box at the right of the tile. You can choose between camera, chirp, chomp, click, clink, coyote, croak, horn, laugh, meow, motor, scratch, silence, splash, warble and sound tiles you create. You can add your own sound by recording it using the SoundRecorder from the Object Catalog.



Moves the object forward in the direction of its heading by the amount of pixels, which is given in the right-most portion of the tile. If you put in a negative number, the object will move backwards. If you use a very large number, your object might disappear from the screen. You can get it back by changing the values in the x- and y-tiles of the object or by choosing "grab this object" in the menu at the top of the viewer.



Changes the heading of the object in degrees by the specified amount. If you use positive numbers, the object will turn clockwise, for negative numbers it turns counter-clockwise.

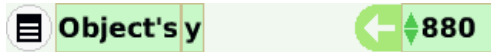


The x coordinate of the object describes the location of the object's rotation center in its holder or the world. The value "0" for the x coordinate defaults to the left side of the object's holder or the world. The default origin of the x-y coordinate system is at the lower left corner of the holder or the world.

You can change an objects rotation center by clicking on it to display its Halo. A small yellow circle with a green arrow will be shown. If you place your mouse on the small yellow circle in the middle of the object, you can hold down the shift key and drag the rotation center to a different part of the object.

Pop Quiz: How can you change the orientation center to make a Baseball bat or Hammer appear to swing?

You can also change the origin point of the World's playfield (or any object with "playfield options..." as a menu item), by selecting "playfield options..." then selecting "origin-at-center". Another way to change a origin point is to click on "set grid spacing..." in "playfield options." The first option allows you to set a new origin point as x@y. These values are relative to the default origin of the lower left corner.



The y coordinate of the object describes the location of the objects rotation center in it's holder or the world. The value "0" for the y coordinate defaults to the bottom side of the object's holder or the world. The origin of the x-y coordinate system is at the lower left corner of the holder or the world.



The heading is the direction the object is facing in degrees. 0 is straight up, 90 is to the right, 180 is straight down and 270 is to the left.

4.2 Category Color



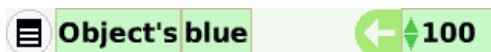
The color of the object. You can change the color by clicking on the color box at the right side of the tile. A Color is made up of multiple attributes as shown in the color category of the viewer. When you set a color all those values are updated. A color picker appears and the color changes to the color you click.



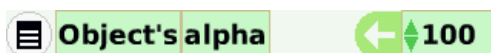
The red value of the object's color from 0 to 100 as a floating point number. You can change the value at the right of the tile.



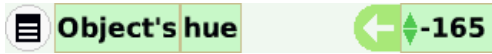
The green value of the object's color from 0 to 100 as a floating point number. You can change the value at the right of the tile.



The blue value of the object's color from 0 to 100 as a floating point number. You can change the value at the right of the tile.



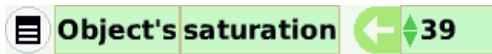
The alpha value of the object's color from 0 to 100 as a floating point number. You can change the value at the right of the tile.



The hue value of the object's color from -180 to 180 as a floating point number. You can change the value at the right of the tile.



The brightness value of the object's color from 0 to 100 as a floating point number. You can change the value at the right of the tile.



The saturation value of the object's color from 0 to 100 as a floating point number. You can change the value at the right of the tile.

4.3 Category Geometry

x

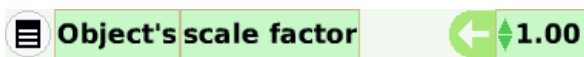
The x coordinate (see the basic category for more detail).

y

The y coordinate (see the basic category for more detail).

heading

Heading is the direction the object is facing (see the basic category for more detail).



The scale factor is the factor by which the object is magnified. It must be greater than zero, but may be less than 1. The maximum value is 10. It can be changed in the box at the right of the tile.



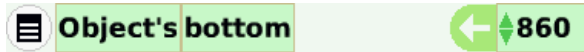
Left is the extreme left edge of the object in pixels. It can be changed in the box at the right of the tile.



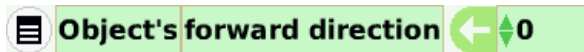
Right is the extreme right edge of the object in pixels. It can be changed in the box at the right of the tile.



Top is the extreme top edge of the object in pixels. It can be changed in the box at the right of the tile.



Bottom is the extreme bottom edge of the object in pixels. It can be changed in the box at the right of the tile.



Forward direction is the angle in degrees of the object's forward direction without rotating it. It can be changed in the box at the right of the tile. If the forward direction is changed, the heading is changed by the same amount. If the heading is changed, the forward direction isn't changed. You can also change the forward direction by clicking on an object to see its halo, then hold down the <shift> key and click on the green arrow and drag it in the forward direction you want to set.

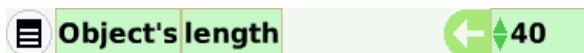
Pop Quiz: Why do you think the designers of Etoys added both "forward direction" and "heading"?

Lets say you draw a Sketch of a person with the head at the top. What will happen when you move it "forward 5"? How can you use forward direction to fix this? Will changing heading fix this?

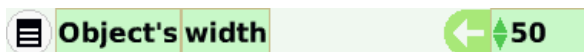
4.4 Category More Geometry



The location is the position of the object, expressed as x,y coordinates in pixels with the @ between the x and y value. It can be changed in the box at the right of the tile.



Length is the greatest length of the object along its original y direction in pixels. It can be changed in the box at the right of the tile.



Width is the greatest width along the original x direction. It can be changed in the box at the right of the tile.



Theta is the angle between the positive x-axis and the vector connecting the origin to the object's position. It can be changed in the box at the right of the tile.



Distance is the length of the vector connecting the origin to the object's position. It can be changed in the box at the right of the tile.



Heading theta is the angle, in degrees, that my heading vector makes with the positive x-axis. It can be changed in the box at the right of the tile.

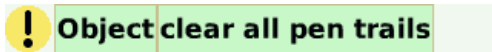


Rotation center x is the x coordinate in pixels of rotation center in parent's coordinate system. It can be changed in the box at the right of the tile.

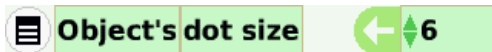


Rotation center y is the y coordinate in pixels of rotation center in parent's coordinate system. It can be changed in the box at the right of the tile.

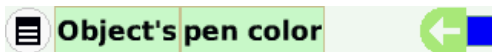
4.5 Category Pen Use



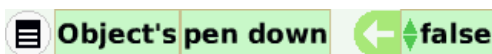
If run, this command clears all pen trails in the object's containing playfield.



This controls the diameter in pixels of dot to use when trailStyle is dots. It can be changed in the box at the right of the tile.



This shows the color of ink used by the pen in the box at the right of the tile. You can change the color by clicking on the color box at the right side of the tile. A color picker appears and the color changes to the color you click.



The pen is down/up if the Boolean choice at the right of the box is true/false. It can be changed in the box at the right of the tile.



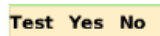
This gives the width of the pen in pixels. It can be changed in the box at the right of the tile.



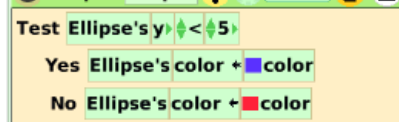
This determines whether lines, arrows, arrowheads, or dots are used when I put down a pen trail. The possibilities can be changed in the box at the right of the tile.

4.6 Category Tests

Getting into the habit of testing your ideas is a very powerful concept. While these tiles are not quite that powerful, they can test what is going on in your Etoys world. You can use the results of these tests to determine the attributes and behaviors of the objects in your world. Getting into the habit of testing your ideas or at least thinking about how you could test your ideas (and what other people tell you is true) is a wonderful "habit of the mind".

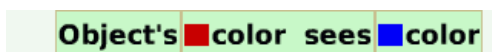


Once this tile is placed in a script, it has the three holders shown below into which you can place one or more tiles:



- **Test:** You can place any tile that contains an attribute of any object in the test area. For example, you could test whether an objects y value is (less than, greater than ,...) a specific number or another objects y value. You can test on almost any attribute (numeric, color, graphic, etc) of any object.
Pop Quiz: How could I define a test to see if a car passed a finish line in a race?
- **Yes:** Tiles placed in the "Yes" holder will be executed when the "Test" is true.
- **No:** Tiles placed in the "No" holder will be executed when the "Test" is false.

Note, that all the tiles below can only be used in the "Test" holder of the "Test Yes No" tile. Some can also be used as watchers, which can be useful when debugging your scripts. While you may think a "Sketch obtrudes" or a "Sketch is under mouse" the Etoys system may think differently. Being able to see this as a script runs can help.




If the object's color (specified by the color on the left side of the tile) is over the given color (specified by the color on the left side of the tile), then the commands under yes in the test will be executed. If not, the commands under no will be executed. You can change the colors by clicking on the appropriate color. A color picker appears and the color changes to the color you click.

Object's is over color  **color**

If any part of the object is over the given color, then the commands under yes in the test will be executed. If not, the commands under no will be executed. You can change the color by clicking on it. A color picker appears and the color changes to the color you click.

 **Object's is under mouse** **false**

If the object is under the current mouse position, then the commands under yes in the test will be executed. If not, the commands under no will be executed.

 **Object's obtrudes** **false**

If the object sticks out over its container's edge, then the commands under yes in the test will be executed. If not, the commands under no will be executed.

Object's overlaps dot

If the object overlaps a given object, then the commands under yes in the test will be executed. If not, the commands under no will be executed. The given object can be changed by dropping a tile for the desired object onto dot.

Object's overlaps any dot

If the object overlaps a given object or one of its siblings or similar object, then the commands under yes in the test will be executed. If not, the commands under no will be executed. The given object can be changed by dropping a tile for the desired object onto dot.

4.7 Category Motion

forward by

See the basic category for a description of this tile.

turn by

See the basic category for a description of this tile.

x

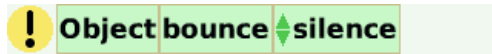
See the basic category for a description of this tile.

y

See the basic category for a description of this tile.

heading

See the basic category for a description of this tile.



If this tile is included in a script, the object will bounce off the edges of its container (ex: world or playfield) and make the sound selected at the right of the tile, when it hits its containers edge.

obtrudes

See the tests category for a description of this tile.



If this tile is included in a script, the object will turn toward the given object. The given object can be changed by dropping a tile for the desired object onto dot.



If this tile is included in a script, the object will wrap off the edge if appropriate.

4.8 Category Fill & Border color

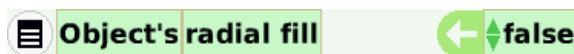
See the color category for a description of this tile.



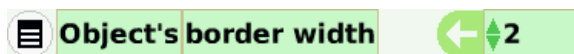
If true is selected at the right of the tile, a gradient fill will be used.



This is the second color used when gradient fill is in effect.



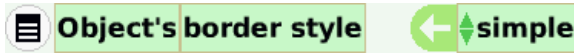
If true is selected at the right of the tile, the gradient fill will be radial.



This determines the width of the border around the object in pixels. It can be changed in the box at the right of the tile.



This determines the color of the object. You can change the color by clicking on the color box at the right side of the tile. A color picker appears and the color changes to the color you click.



This determines the border style. You can select from simple, raised, inset, complex framed, complex raised, complex inset, complex alt framed, complex alt raised, and complex alt inset at the right of the tile.



If true/false, the corners are rounded or not. Select true/false at the right of the tile.

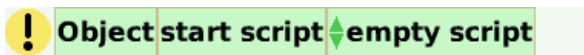


If true/false, a drop shadow is shown or not. Select true/false at the right of the tile.



This is the color of the drop shadow. You can change the color by clicking on the color box at the right side of the tile. A color picker appears and the color changes to the color you click.

4.9 Category Scripting



This starts the given script ticking. You can select from available scripts at the right of the tile.



This makes the given script be "paused". You can select from available scripts at the right of the tile. Note: If the script specified is the script containing this tile, the script will not pause until it finishes its current processing of the all the tiles in the script. For example, if you have a "Object forward 5" tile after a "Object pause script script1" tile in a script named "script1", the object will move forward 5.


 **Object stop script**  **empty script**

This makes the given script stop or revert to "normal". You can select from available scripts at the right of the tile.

Note: If the script specified is the script containing this tile, the script will not stop until it finishes its current processing of the all the tiles in the script. For example, if you have a "Object forward 5" tile after a "Object stop script script1" tile in a script named "script1", the object will move forward 5.

 **Object start all**  **empty script**

This starts the given script and all of its siblings's scripts ticking in the object. You can select from available scripts at the right of the tile.

 **Object pause all**  **empty script**



This makes the given script and all of its sibling's scripts be "paused" in the object. You can select from available scripts at the right of the tile.

Note: If the script specified is the script containing this tile, the script will not pause until it finishes its current processing of the all the tiles in the script. For example, if you have a "Object forward 5" tile after a "Object pause all script1" tile in a script named "script1", the object will move forward 5.



 **Object stop all**  **empty script**

This makes the given script and all of its sibling's scripts stop or revert to "normal" in the object. You can select from available scripts at the right of the tile.

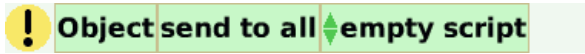
Note: If the script specified is the script containing this tile, the script will not pause until it finishes its current processing of the all the tiles in the script. For example, if you have a "Object forward 5" tile after a "Object stop all script1" tile in a script named "script1", the object will move forward 5.

 **Object tell all siblings**  **empty script**

This sends a message to all siblings to run the given script once. The object that executes this will not have its script run, only its siblings will run the script once. You can select from available scripts at the right of the tile.

 **Object do**  **empty script**

This runs the given script once, on the next tick. You can select from available scripts at the right of the tile.



This runs the given script in the object and in all of its sibling once. You can select from available scripts at the right of the tile.

4.10 Category Sound

make sound

See the basic category for a description of this tile.



This will play a sound of the frequency entered in the box at the right of the tile.



This command will stop the sound.

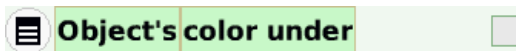
4.11 Category Observation



This tile goes into a test box to test the heading the object would need to have to face directly toward another object. The given object can be changed by dropping a tile for the desired object onto dot.



This tile returns the brightness under the center of the object. The brightness appears in a box at the right of the tile.



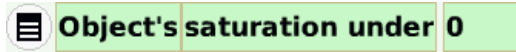
This tile returns the color under the center of the object. The color appears in a box at the right of the tile.



This tile goes into a test box to test a function of the distance to another object. The function can be selected at the left of the tile once it is in the test box. The given object can be changed by dropping a tile for the desired object onto dot.



This tile returns the luminance under the center of the object. The luminance appears in a box at the right of the tile.

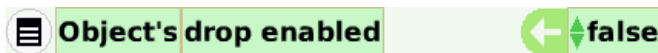


This tile returns the saturation under the center of the object. The saturation appears in a box at the right of the tile.

4.12 Category Drag & Drop



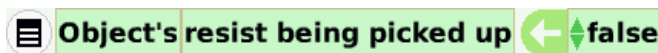
If true/false, this object will be blind/or not to all input. Select true/false at the right of the tile. This is useful if you have text or other objects you don't want the user to accidentally move or change.



If true/false, drop is enabled/or not. Select true/false at the right of the tile. When true you can embed other objects in this this object. Once embedded when you move the object all objects embedded inside it will move with it.

Etoys Challenge:

1. Use drop enabled to draw a person and ensure "drop enabled" is true.
2. Then draw some clothes (some combination of shirts and shorts).
3. Place the person and each piece of clothing in a "Maker Button" (this will allow you to make copies instantly just by clicking on the maker button and dragging onto the screen).
4. Then see how many possible combinations of different outfits you can create.
5. How can you prove you have all the possible combinations?



If true/false, a simple mouse-drag on this object will allow it to be picked up/or not. Select true/false at the right of the tile.

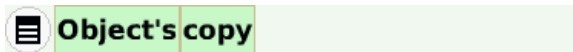
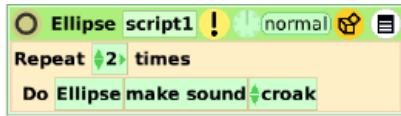


If true/false, the object is resistant to easy removal via the pink X halo handle/or not. Select true/false at the right of the tile.

4.13 Category Miscellaneous

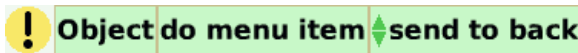
Repeat Times

Drag here to tear off a Repeat/Times unit which you can drop into your script. This tile is also available in the scriptor. This Tile will execute all the tiles inside it the number of times specified as shown below.



This tile makes a copy of the object. The copy will not be visible, you need to include it in a playfield or holder first. An example would be "playfield include: object's copy". A difference is that for siblings, "copy" creates another sibling.

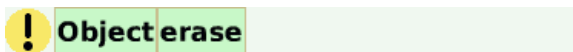
The menu at the left of the tile allows you to get a piece of data from the object which you can use in scripts. Also you can immediately make a copy of the object.



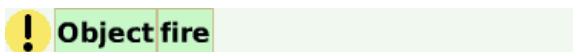
This tile allows you to do the menu item selected at the right of the tile. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



The number at the right is the object's index in its container (holder or world). The index can be changed at the right of the tile.



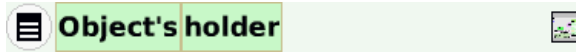
This removes this object from the screen. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



This tile triggers any and all of this object's button actions. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



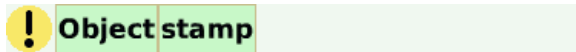
This tile makes the object invisible. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



This tile shows the object's container at the right of the tile. The menu at the left of the tile allows you to get a simple or detailed watcher or piece of data from the object's holder.



This tile makes the object visible. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



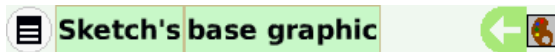
This tile adds the objects image to the pen trails. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



This tile adds the object's image to the pen trails and then goes away. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



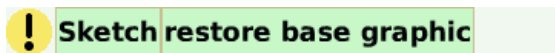
This tile shows the picture currently being worn at the right of the tile.



This tile shows the picture originally painted for this object at the right of the tile, but can subsequently be changed via menu or script.



This tile allows the object to wear the costume of a second object. The second object can be changed by dropping a tile for the desired object onto dot. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



This command restores the object's costume to the one I remember in my baseGraphic. You can either click the yellow exclamation mark at the left of the tile or use the tile in a script.



This tile allows the user to select how the graphic should change when the heading is modified. It can be used in a test box to test for the rotation style.

4.15 Category Input



This tile can be used in a test box to test for last unhandled keystroke.

Common Menu Items

4.16 Common Menu Items

send to back - this puts the object behind other objects.

bring to front - this puts the object in front of other objects.

embed - this allows you to embed the object inside another object, for example you could embed All Scripts in a playfield, book or in a particular page of a book show it only shows on that page. The options only show when overlaying another object. When selected you will be given a list of objects into which you can embed this object.

change color - this option when clicked opens up the medicine dropper and paint palette tool. Clicking a color from the color palette changes the color of the background of this object.

border style: this option sets the style of the border based on color, width and style.

- border color - brings up the medicine dropper and paint palette tool. Clicking a color from the color palette changes the color of the border of this object.
- border width - brings up a tool that sets the thickness of the border. Drag the tool around and choose the width
- simple
- inset
- raised
- complex alt framed
- complex alt inset
- complex alt raised
- complex framed
- complex inset
- complex raised

drop shadow: this gives the user the option to set the color of the shadow, visibility, and the shadow offset.

resist being deleted - when this is set, the object can not be deleted with the "x" button from its halo. The default value is unset.

resist being picked up - when this is set, the object can not be picked up by the pick-up and move tool on the halo. The default value is unset.

be locked - when this is set, the user can not affect any changes to the object.

provide clipping -whether the parts of objects within this object that are outside its bounds can be masked.

direction arrow - when set, the direction arrow will appear on the center of the halo. The default value is unset.

accept drops - when set, allows objects to be dropped into the *all script* body. The default value is clear.

round corners - when set, makes the corners of rounded. Default is not set.

siblings :

- make a sibling instance - makes another morph who's player is of the same class as this object. Both siblings will share the same scripts.
- makes multiple siblings - makes multiple instances.
- indicate all siblings - make all siblings visible.

extras: gives the user the following options

- adhere to edge - default is none, which means the object can be placed anywhere in its container. Other options let you specify if the object sticks to an edge, corner or center of its container.
- draw new path
- follow existing path
- delete existing path
- balloon help for this object - allows you to change the balloon help for an object. The balloon help will show when you leave the mouse over an object for a period of time.

License: *The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, you agree that your edits can be used under both the GPL and MIT licenses.*

5. Objects

In this chapter you will find descriptions for all objects available for use in Etoys. You can find these objects in the supplies bin or in the object catalog. The object catalog itself is also part of the supplies bin. Objects are organized in categories, some objects can be found in more than one category. Every object can be found by clicking "find" in the object catalog, and then typing the name of the object.

The objects in this chapter are listed in alphabetical order.

All Scripts



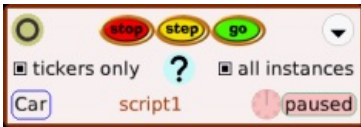
The All Scripts object has a stop, step and go button for controlling all the scripts at once.



When you click on the round left icon, the object will be closed.



When clicking on the right icon with the down arrow, the tool will be "opened up" so users can control each script in the project individually.



This pauses all running scripts.



This button runs every posted scripts exactly once. To run the scripts continuously using this button, keep it depressed with the mouse. The scripts will continue running until you release it with the mouse button.



When this button is pressed once all the scripts will run continuously.

tickers only - The user can choose whether to just show scripts that are paused or ticking by setting this option or show all scripts by unsetting it. The default value for this option is "set".

all instances - If set, then entries for all instances will be shown. If deselected, only one representative or a group of similar objects will be shown. This option is "set" by default.

Arrow

An arrow is a line with an arrowhead. It can be used as a tool to draw arrows for illustrations. The Arrow Object can be found in the Objects Catalog under "Graphics". Drag the arrow and place it in your playfield. The default state of the arrow doesn't show its handles. To make the handles visible, please refer to the Unique Menu Options below for a description.

Arrow handles mark the beginning and end points of the *Arrow* object. Do not overlap the beginning point (little blue circle handle - also signifies the cursor) and its end point (little yellow circle handle) to alter its state from a line to a point. This will create an unrecoverable error. Users would essentially have to "start over" and drag and drop another *Arrow* object.

Unique Tiles

Aside from the common tiles found in other objects, the *Arrow* object has unique tiles under the Polygon and Input Category.

vertex cursor - A vertex is defined as a point of intersection between two lines. as mentioned above, the vertex cursor is indicated by the small blue circle handle. The default location of this cursor is in location "1" for a simple arrow (also the beginning point). To move this cursor to a different vertex point, click the arrows or input the vertex number.

vertices count - Vertices is the plural noun for a vertex. This tile indicates the number of vertices of the polygon.

x at cursor - this tile indicates the x coordinate of the cursor.

y at cursor - this tile indicates the y coordinate of the cursor.

add a vertex at the beginning - click this once or use in a script to add vertices in the beginning of the vertices list (vertex location 1). Click and drag the small yellow circle in location one to reveal the added vertex (or vertices).

insert a vertex at cursor - click this once or use in a script to add vertices in the cursor location (refer to the vertex cursor tile). Click and drag the small blue circle handle to reveal the added vertex (or vertices). *add a vertex at end* - click this once or use in a script to add vertices at the end (refer to the vertices count to reveal the last vertex number). Click and drag the small yellow circle in location the end location to reveal the added vertex (or vertices). *remove all vertices but cursor* - this deletes all of the vertices but the cursor.

remove the vertex at cursor - this deletes the vertex at the cursor location.

shuffle vertices - randomly switch vertices positions.

line is curved - this will not curve a standalone arrow but it will curve the lines when the arrow is used to create a polygon.

line is opened - A user can create a closed polygon by not only closing the points or vertices but also by simply changing this value to "false".

showing handles - the default value is "false" where the handles/vertices are not shown.

Unique Menu Options

show handles - same functionality as the tile "show handles". The default value is unset. To show handle, click and set this button value.

closed - this closes the loop of the polygon.

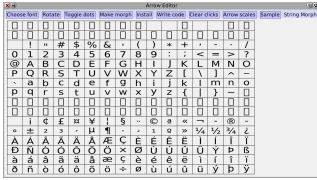
curved - this curves the sides of the polygon.

specify dashed line - this creates different dash styles for the lines of the arrow or polygon sides.

You can specify the location of the arrowhead by selecting the different options in the menu: none (no arrowheads), arrowhead at the beginning, arrowhead at end vertex, arrow heads on both the beginning and end vertices.

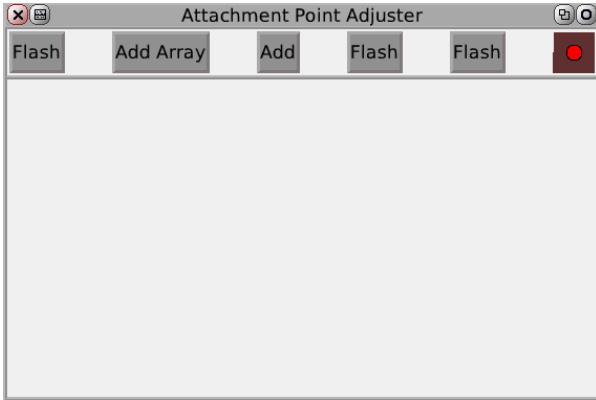
customize arrow - You can chose to use the standard arrowhead or customize the look of the arrowhead by clicking this option. This option is only useful if you already have an existing arrowhead. It will launch a tool that you can move around to change the look of the arrowhead.

Arrow Editor



TBD VOLUNTEER NEEDED TO WRITE THIS SECTION

Attachment Adjuster



TBD VOLUNTEER NEEDED TO WRITE THIS SECTION

Badge

A tool for collaborating with other Squeak users

TBD VOLUNTEER NEEDED TO WRITE THIS SECTION

Book



The book is a multi-paged structure. It is an authoring object that can be used for both static and dynamic illustrations and presentations. To use the *book* drag it from the supplies bin into your playfield. Text and other objects can be dragged into the pages of the *book*. Aside from the common tiles the book has special tiles for navigating pages and getting the last keystroke entered.

The book can also be described as a multi-paged playfield. Users who have the experience in using the *playfield* object would find that each of the book's pages are in themselves objects and are structurally the same as a playfield. Each page of the book has the same options and tiles that can be found in the *playfield* object.

The book has its basic controls and short menu options on top of a single page by default. The page controls come in two formats basic (short) and advanced (long) Although the default controls include a couple of navigation arrows it won't be useful unless you access the more advanced controls and add pages.

Basic Controls (Short)



The basic controls is the default format. It comes with two navigation arrows. One to move forward one page and one to move backwards one page, a menu, page indicator with total pages value and a button to switch to more advanced controls.

Menu Options (Short)

find - click this option to search for words within the pages. The words are not case sensitive and don't have to be complete. This option also lets you find parts of the words.

go to page - type in the ordinal number of the page you would like to navigate to. If the page number that you typed in exceeds the total number of pages, Etoys will not give you an error but you will remain on the page where you initiated the request.

show more controls - aside from the button on the left most of the controls panel, this option allows you to get expanded format (advanced option).

revert this page - to use this option, you must first switch to the advanced format of the controls panel and click the "save the page for later revert" option. This allows you to go back to the original state of your page.

revert entire book - As with the *revert this page* option, one can also save the entire book for a later revert.

Advanced Controls (Long)



In addition to the basic controls you got two icons with double arrows to navigate directly to the first resp. the final page. The + allows you to quickly add one page after the current page, the - will delete the current page. The advanced controls also have additional menu items.

find - same functionality as described in the basic controls format.

find again - this gives you the option to search the next instance of the text that you are trying to find.

duplicate this page - this makes a copy of the current page and inserts that copy as the next page.

revert this page - same functionality as described in the basic controls format.

revert entire book - same functionality as described in the basic controls format.

keep all pages the same size - when this option is set [black] all the pages follow the same dimensions of the current page where this option was selected. The default of this option is clear.

page controls at top - this button is set by default. When deselected, the control panel moves to the bottom of the page. This will be true for all the pages of the book.

page control short - within the advance control menu this button is deselected. Select this button to get the basic controls format.

page controls visible - this button is set by default. Deselecting this button will cause the controls panel to disappear. To retrieve the controls panel, bring up the book's halo by right-clicking the book. Select the Menu icon and scroll down and click the "book.." option. This will take you to the the advance controls menu where the *page controls visible* option is available again.

scripting area - when selected if you have any "stop, step, go" buttons in the book will only control the scripts of objects inside the book. Also when selected, if you open the viewer of an object in the book, the viewer will open inside the book.

view pages full screen - If set, all the book's pages occupies the entire screen. The toolbar is hidden from view and the book's control panel minimizes and moves to the top left of the screen. To exit the full screen mode and make the toolbar visible, click on the book's menu and select this button again.

wrap after last page - This options is set by default. If deselected, the book will not cycle to the first page after the last page is reached.

sort pages - when selected, it opens up a small window/tool where one can arrange the pages of the book.

hand me a bookmark for this page - this creates a bookmark button for the page. Drag the created bookmark button outside of the page and change the label to make it distinctive to the page (follow instructions on *Button* object on how to do this). The button takes you back to the bookmarked page.

hand me a thumbnail for this page - this creates a small icon to represent the page.

visual and sound effects:

set sound effects for this page - clicking this option opens up 18 choices for sound effects for the page. The sound plays as one navigates to the page. The user can also his/her own sounds. See the sound recorder object for more information.

set visual effects for this page - clicking this option gives the user 11 different visual effects (including "none") for the page. The visual effects are visible upon turning the page. Only one visual effect is active per page and each page can have a unique visual effect.

set sound effects for all pages - clicking this option opens up 18 choices for sound effects for all of the pages of the books. The sound plays as one navigates to each page of the book. The user can also his/her own sounds. See the sound recorder object for more information.

set visual effects for all pages - clicking this option gives the user 11 (including the default "none") different visual effects for all the pages of the book. The visual effect chosen will be visible upon turning each page.

mark this page to be revertible - this saves a snapshot of the page once at its current state. Users can then go back to this state by going back to the menu and selecting "*revert this page*" option (see the section on this option above for more information). Selecting this again will take another snapshot that will overwrite the last one.

mark entire book to be revertible - this saves a snapshot of the entire book once at its current state. Users can then go back to this state by going to the menu and selecting "*revert entire book*" option (see the section on this option above for more information). Selecting this option again will take another snapshot that will overwrite the last one.

advanced:

make all pages the same size as this page - clicking this option makes all the pages of the book to be the same sizes as the current one.

set background color for all pages - selecting this option launches the medicine dropper tool which allows a user to chose the color of the page from the color palette. Selecting a color sets the background color for all of the pages of the book.

uncache page sorter - When you have clicked "sort pages" before in your book options, you will get a list of all your pages with thumbnail pictures for easier sorting. This option will delete the thumbnail pictures.

make a thread of projects in this book - this option takes all projects you have put in the book, puts them in a thread and adds a project navigator to the project. This allows you to navigate easily through all these projects. But this option has a bug in Etoys 4.0. You will only get the navigator for the projects from your book, and the book project itself is not included. You will not be able to return to your book project using the navigator or the arrows in the toolbar. The only way back is to click ALT-SHIFT-W, choose "Jump to project" and then choose your book project.

make this a template for new pages - choosing this option allows newly created pages to have the same properties as the current one. This option behaves exactly

show full screen - If selected, all the book's pages occupies the entire screen. The toolbar is hidden from view and the book's control panel minimizes and moves to the top left of the screen. To exit the full screen mode and make the toolbar visible, click on the book's menu then *advanced* option then select "exit full screen" option.

Unique Tiles

goto - go to the given page. This tile does not accept numbers but accepts the tile representation for the page. To get the tile from the page, bring up the page's halo and click the orange button.



Drag this tile to replace the default value "dot".

next page - Unlike the *goto* tile where one can jump from one page to any pages in the book, the *next page* tile only takes the user one page forward.

previous page

Like the *next page* tile where one can jump one page forward, the *previous page* tile takes the user one page backwards.

first page - This tile takes the user to page 1 of the book.

last page - This tile takes the user to the last page of the book.

page number - The ordinal number of the current page.

number of pages - Indicates the total number of pages that the book contains.

page controls showing - The page controls are by default on top of the book. Using this option, you can choose if you want to show or to hide them. A "true" value on the *page controls showing* tile says "yes I want the controls to show". A "false" value will hide them.

page controls short - Whether page controls are shown in short form or in long form.

page controls at top - If "true", page controls are shown at the top of the book, if "false", they will be shown at the bottom.

revert page - revert to the original version of this page.

revert all pages - revert the entire book to its original contents.

Bouncing Atoms

An object with dots (atoms) that bounce off each other and off the walls of the box. From the bouncing atom's menu, you can start infection, set atom count and show infection history. The blue atoms are replaced by red atoms as they are infected once you start the infection and you can watch the infection grow in a graph if you select show infection history from the menu. The atoms stop after you set the atom count and they restart if you click the bouncing atoms object.

Broom

The broom allows you to align objects vertically or horizontally (depending on the direction first used to drag the broom). To align objects simply click on the broom and drag to align the objects. Once you start dragging the broom will change into a line. The broom's size can be extended by dragging along the line shown when you first start dragging the broom.

Button

Press me This is a button to use with tile scripting. Its script will be a function of its containing playfield. To use the Button object, drag it out of the Supplies bin and into the playfield where you're going to use it. In its default state, the script for the button is empty. To add scripts to this button, bring up its halo by right-clicking it. Click on the green button with a rectangle on it to bring up a script window. A script for its containing playfield will be created. NOTE: If you move the button to another playfield it will reference a script in the new playfield and have a different behavior.

You can also create a button by first creating a script then clicking on the menu in the Scriptor and selecting the "button to fire this script"-option from the script window. This button will not lose its reference

Buttons are controls that users click to make something happen. The default Button displays text inside a rounded rectangle.

To create text that responds to user mouse-clicks, set the border width of the Button to zero, and the Button color property to the same color as the Button's holder or container (for example, the page it is on). The border width property, the color property, and the label property are located on the Button pane of the Button Viewer.

To change the font of a Button, click the Menu button in the Button's halo. Select "Unlock String". Right-click on the Button's label twice. The first right-click will display the Button's halo, the second right-click will display a halo for the String object that contains the Button's label. Click the Menu button in the String's halo, and select "change font".

To create a picture button, create a Sketch using the Paint tool, and add a script to the Sketch object that fires on a Mouse Down event.

Unique tiles

label - You can get or change the wording on the button.

color

You can get or change the color of the button.

height

You can get or change the height of the button.

border color

You can get or change the color of the button's border.

border width

You can get or change the width of the button's border.

height

You can get or change the height of the button.

rounded corners

You can say whether the corners of the button should be rounded or not.

act when

You can choose if the script should fire when the button is pressed, up, or down.

fire

Trigger any and all of this object's button actions.

Button Bar



TBD VOLUNTEER NEEDED TO WRITE THIS SECTION - NOTE: Couldn't figure out purpose of this, other than when clicked it "looks like a button being pressed"

Button Down?>

This is a tile for querying whether the left-clicker of the mouse button is down. This tile is also available within the *scripts* tile window.

Button Flap

The Button Flap object creates an interface element that works like a pull-out drawer in which you can store other objects. When the Button Flap's tab is clicked, the drawer "opens", revealing a "parts bin". The parts bin is a Playfield object. Clicking the Button Flap's tab again hides the parts bin.

Button Flap settings are controlled via the object menu on the flap's halo.

New objects can not be dragged from the Object Catalog directly onto the parts bin. Instead, first create the object in your project, and then move the object to the Button Flap's parts bin.

Button Flaps may be used to create "About" or "Read Me" information. In Squeak, Button Flaps are used as tool drawers. Each flap contains a family of tools.

Button Flap settings (changed via the Button Flap menu) include:

tab color

The color of the Button Flap's tab

flap color

The fill color of the flap's parts bin.

cling to edge...

Sets the location of the flap (top, left, bottom or right)

change tab wording...

Changes the text that displays on the tab

use graphical tab

Displays the tab as a graphic instead of the default text. Once this option is selected, a new **choose new graphic...** option appears on the menu.

use solid tab

Displays a solid colored tab with no text.

parts-bin

NOTE:SEEMS TO HAVE NO Affect, also there is the menu for the "Flap" handle, versus the menu for the "Flap"

pop out on dragover

NOTE: I could not detect any differences when selected or not

pop out on mouseover

When selected the "Flap" will expand when the mouse touches the flap handle. When not selected you need to click on the flap handle.

shared by all projects

When selected the flap will be displayed in all projects until you exit Etoys

BUG: - if you set to shared by all projects and then switch to another project and back using Previous Project/Next Project (or project icons), a second instance of the flap is created.

compact flap

When selected the length of the flap will match that of the side to which it is attached. When unselected the flap size, will minimize to the smallest possible size that can contain its contents.

destroy this flap

Another way to delete the flap.

Button Up?

This is a tile for querying whether the left-clicker of the mouse button is up. This tile is also available within the *scripts* tile window.

Camera

A player for video devices like cameras, video capture cards, etc. Camera works automatically with the XO computer. (*Can someone describe how to make camera work with a Mac or PC?*) From camera's menu, you can configure the video device (see below) or start showing statistics (in transcript). Once you show statistics, you can reset them from the menu. (*Can someone add more about showing statistics?*) Camera has a special tile in the graphics category

- **camera's graphic** is the picture currently being shown.

and 9 tiles in the special viewer category video.

- **camera play** is a command to start the camera.
- **camera stop** is a command to stop the camera.
- **camera's is running** is a tile that returns true/false if the camera is/is not running.
- **camera's auto extent** sets the height of the slider. This can also be set using the yellow change size button on the slider's halo.
- **camera's brightness** allows changing of the camera's brightness.
- **camera configure video device** allows the user to set the frame width and height, test pattern (lenna or color bars) and choose to show the time or not in the camera window.
- **camera's contrast** allows changing of the camera's contrast.
- **camera's last frame** is a morph with the last frame. (*can someone describe this more completely?*)
- **camera's resolution** is a tile that allows you to pick among original, 256 colors, 256 grays, 4 grays and black and white.

Chess

This is a chess game which you can play against the computer.

Chinese Checkers

This is a Chinese checkers game which you can play against the computer.

Cipher

The Cipher Panel displays an encrypted message. The encryption is a simple substitution code; each letter of the alphabet has been changed to a different one. You can solve the cipher by clicking above any letter in the message, and typing the letter you think it should be. The Cipher Panel automatically makes the same substitution anywhere else that letter occurs in the encoded message. If you are having trouble, you can use the command menu to 'show cipher hints'. That will display how many of each letter occurs, which is often a help in solving ciphers.

Circle

A circular shape object. This object can also be created using the Ellipse object (with some measure of manipulation).

Clipboard

This object will always show whatever is on the text clipboard

Clock

A ticking clock with hour, minute, and second hands. The time on the clock is the time on the computer. From the clock menu, you can choose roman numerals and anti-aliasing, change font, change hands color and change center color.

Column

An object that presents the things within it in a column A basic Connector that bends smoothly

Connector

Connectors are lines (straight or curved, with or without arrowheads) that can be used to show connections between objects. They stay connected to the objects when the objects are moved around (so you don't have to redraw the lines each time you rearrange your objects). You can change the shape of a connector by clicking on a point in the line and dragging that point to a different area of the screen. This will add a new vertex or curve to the line.

Connectors can be used for:

- diagrams
- flow charts
- mind maps
- showing cause and effect (see Intel's Seeing Reason Tool)
- showing the relationships between ideas

Connectors also have a "hidden" power. Once an object is "connected" a new Viewer-Tile-Category appears for the connected object called "connections to me", This allows you to run scripts (using the tell all ... tiles) on items connected to an object.

The following are the different types of connectors that are in the object catalog:

- Connector - a basic connector with no arrows at its ends
- ConnectorArrow - a basic connector with and a basic arrow at its destination point
- Curvy Connector - a basic connector with its "smooth curve" attribute set to true
- Curvy ConnectorArrow - a basic connector with its "smooth curve" attribute set to true and a basic arrow at its destination point
- Schematic Connector - *same as a Connector as far as I can tell, did not see any differences in attributes, seemed to behave differently in the way they attach to objects (such as rectangles, I saw they tend to connect or switch connections to vertices more easily than "Connectors", but the ability to choose connection points, IMHO should be independent of the type of connector and should not change when moving connected objects around.*

- Random Connectors - a button that will create random arrows each time you drag one from it.

You can also set the arrow heads and the size of arrowheads from Menu icon in the Connectors Halo:

- arrow sizes - changes arrow heads size by dragging
- add label - will add a text label that will move with the arrow. You can change the position of the label relative to the connector by dragging the label to a different position.

Exploration Challenge: Drag out multiple "Random Connectors" and look at the attributes in its "connector" category. See if you can figure out what attributes to change to create your own special connectors. Then if you like drop them into a maker button or a Button Flap for re-use.

Connector Arrow



A basic connector with and a basic arrow at its destination point

Connector Broom



The broom (which applies to all objects not just connectors) allows you to align objects vertically or horizontally (depending on the direction first used to drag the broom). To align objects simply click on the broom and drag to align the objects. Once you start dragging the broom will change into a line. The broom's size can be extended by dragging along the line shown when you first start dragging the broom.

Connector Button



A maker button with a "Connector Arrow" loaded into it. Each time you click on the button you create a new "Connector Arrow".

Connector category: geometry has additional tiles

total length

The total length of the connector.

midpoint x

The X coordinate of the connector's midpoint

midpoint y

The Y coordinate of the connector's midpoint

source x

The X coordinate of the connector's source object

source y

The Y coordinate of the connector's source object

destination x

The X coordinate of the connector's destination object

destination y

The Y coordinate of the connector's destination object

Connector category: color & border

line width

The width of the main part of the Connector.

new category: connected to me

This category is added to the objects at the source and the destination of the connector.

incoming connection count

The number of connectors whose destination end is connected to this object

outgoing connection count

The number of connectors whose source end is connected to this object

tell all incoming connections

Send a message to all the connectors whose destination end is connected to this object. If the connector has a script name that matches the message that script will be executed once.

tell all outgoing connections

Send a message to all the connectors whose source end is connected to me. If the connector has a script name that matches the message that script will be executed once.

tell all predecessors

Send a message to all graph predecessors. This will send a message to the object(s) at the other end(s) of incoming connector(s). If the object has a script name that matches the message that script will be executed once.

If the connector has a script name that matches the message that script will be executed once.

tell all successors

Send a message to all graph successors. This will send a message to the object(s) at the other end(s) of outgoing connector(s). If the object has a script name that matches the message that script will be executed once.

new category: connector

source

The object at my first end.

destination

The object at my second end.

orthogonal

If true, the connector shape is made of horizontal and/or vertical segments.

If false, the connector line can be made up of straight lines.

smooth curve

If true, the vertices of the line are rendered as a smooth curve connecting the line segments.

If false, the vertices of the line are rendered as a points connected the line segments.

NOTE: Vertices can be added to any connector by clicking on any point in the line and dragging the point to form a new vertex.

source arrow name

The name of the arrow at the source end of the connector.

destination arrow name

The name of the arrow at the destination end of the connector.

NOTE: Valid names I have discovered so far are: #noArrow, #basicArrow, #test2, #umlArrow

DEVELOPERS: Please provide complete list.

straighten

If the Connector's attribute "orthogonal" is false, Change the shape to a straight line .

If the Connector's attribute "orthogonal" is true, the connector's shape will not change.

BUG: "straighten" can change the shape of an connector when orthogonal is set to true. Steps to recreate:

1. Drag two rectangle onto the screen from the supply bin diagonally opposite each other.
2. Connect them with a connector from from the object catalog
3. Change the connectors "orthogonal" attribute to True (this will change the shape of the connector vertical line from the source to a horizontal line to the destination, and L shape)
4. Click on a point on the vertex in the connector and drag to "invert" the L (ie: from lower left to upper right)
5. Run the command "straighten" on the connector. This will revert the shape of the connector back to the "L" shape.

add label

Add a label at the middle of the connector. You can add multiple labels to a connector.

remove labels

Remove all labels associated with the connector.

Connectors Flap



NEEDS TESTING: There are some bugs here (was not able to consistently reproduce and I am tired, but definitely this needs more testing. I got some "Abandon" booms a few times.

Croctic

The Croctic Panel presents an acrostic puzzle for solution. As you type in answers for the clues, the letters also get entered in the text of the hidden quote. Conversely, as you guess words in the quote, those letters will fill in missing places in your answers. In addition, the first letters of all the answers together form the author's name and title of the work from which the quote is taken.

Curve



A smooth wiggly curve, or a curved solid. Shift-click to get handles and move the points. By clicking the menu button on the halo, curve's handles can be shown, curve can be made "open" rather than closed, curve can be converted into a polygon by unchecking curve, and the perimeter of curve can be changed from solid to a variety of dashed lines. Curve has the same color category as rectangle.

Curvy Arrow

A curved line with an arrowhead. *{The vertices of this object can be manipulated using the same methods in the Polygon object please see ObjectCatalogBasic}.*

As with the arrow tool, putting two vertices together deletes one of them. Do not attempt to do this to close the loop.

Curvy Connector



A basic connector with its "smooth curve" attribute set to true

Curvy Connector Arrow



A basic connector with its "smooth curve" attribute set to true and and a basic arrow at its destination point

Digital Clock

A digital clock that displays the hour on the computer. From the clock's menu, you can choose to change font, show seconds and display 24-hour. The digital clock has the special viewer category with 3 tiles.

- **Digital Clock's hours** returns the digital clocks numeric hour.
- **Digital Clock's minutes** returns the digital clocks numeric minute.
- **Digital Clock's seconds** returns the digital clocks numeric second.

DoubleClick

An example of how to use double-click in moprhic

Ellipse



An elliptical or circular shape. To make it a circle, set the length and width the same in the more geometry category of its viewer. Ellipse has the same color category as rectangle.

Enhanced Text



The "Enhanced Text" object differs from "Text" object in that it has an extra attribute in the Viewer's "text" category called "focused". "focused" is set to true when the cursor is inside the text object. The **focused** attribute can be used to prompt the user for input.

Enhanced Text also has two additional menu items that "Text" does not:

- **accept on CR:** this attribute will cause the focus to be shifted to the next object when the user hits the <Enter> or <Return> key. The next object will be the next text object (as determined by the "element number" attribute in the Viewer's "miscellaneous" category).
- **accept on focus loss:** which (HELP couldn't figure out what this does)

Enhanced Text does not have the following menu items **translatable** or **use pango** both of which are found in Text.

Event Theatre

A framework for creating tutorial snippets. The Event Theatre provides a framework for authoring "event-movies". It uses custom variants of the Navigator, the Supplies flap, the painting system, property sheets, Viewer flaps, etc., all of which reside within the controlled confines of the Theatre. To author an event-movie, get a new Event Theatre from the Objects catalog.

1. Resize the Theatre, using the halo, to the delivery size desired for playback.
2. Use the menu in the controls panel to add or delete Supplies and Navigator flaps as desired.
3. Edit the "caption" by clicking on the text that says "Untitled" and typing your desired caption. This caption is affixed to playback buttons, and, generally, provides a way to identify the event-movie.
4. Set up the "initial conditions" for the event-movie you're about to record (e.g., paint a background, provide some explanatory text, add objects you want to be present at the start of the movie.) When the recording is first opened by the user for playback, and whenever the recording has been rewound, this is exactly what it will look like.
5. When ready to start the recording, press the "Record" button. A red border will be seen around the recorder, to indicate that recording is in progress. Recording will continue until you hit the ESC key.
6. Note that if you want to include sound in your recording, you can add it directly during playback, or you can produce voiceover externally and add it in later using the Event Roll.
7. Now proceed to "do", with the mouse and keyboard, whatever you wish to record. For best results, all mouse gestures should be made within the interior of the Theatre.
8. Hit ESC when done recording.
9. To review what you've recorded, press "Play". If unhappy with the result, repeat steps 1-8.
10. If you're happy with the result, and now wish to add a sound, open the sound panel, then click Play to replay the recording, and whenever you wish to add a snippet of voiceover, click on the "Start Recording Voiceover" button, and start talking, and when done with that snippet click the "Stop Recording Voiceover" button. Once the playback finishes, the added voiceover(s) will become part of the event tape, and will be seen in the event roll.
11. When you're happy with the result, hit the "Publish" button, to get a playback button. There are currently two choices:
 - a. Iconic Button - Initially provides a picture of the initial scene of the movie, scaled 0.3x, and overlaid with the word HINT. When the user clicks on such a button, the event-movie is played back in an ephemeral "playback theatre", and after the playback is done, the playback theatre shrinks down to a 0.5x scale-downed picture of the **final** scene of the movie. Subsequent hitting of the button will again invoke a playback.
 - b. Textual Button - a simple labeled button which, when pressed, triggers playback of the event tape.

12. The playback button you obtain when you "Publish" can be placed anywhere, such as on the page of a book. You can control, via a playback-button's halo menu, whether or not it should be "auto-start", and whether or not "auto-dismiss". When the user presses the button, a "Playback" space will open, which resembles an Event Theatre, but has only playback-relevant controls. A playback set up for both auto-start and auto-dismiss comes without any controls.
13. To edit the "event tape" of a recording you have made in an Event Theatre, and for a generally good time, click on the interlocking-circles icon to obtain a tool that allows you to visualize and to edit a "score" or "piano roll" of the event tape.

Summary of terms

Event Theatre

The main tool for creating an Event Tape.

Event Roll

An auxiliary tool showing the full "score" of an Event Tape.

Event Tape

The results of an event-theatre session; an interaction sequence that can be played back.

Event Recording

A term interchangeable with "Event Tape."

Event Movie

What you see when you play back an Event Tape.

Dragging out an event player produces a mini world where you can record all your actions for later play back. The text above is from the excellent help guide that can be accessed through the question mark at the lower left of the object.

File Dialog

The File Dialog box lists all the directories and files within a user's system. The tool can find files in directories and group them by type: Art (.gif, .png), Morphs (.morph), and Projects (.pr). There are buttons to load the project, make an XO bundle and to cancel the search operation.

Fish Eye

An extreme wide-angle lens

Guides

A tool for sending objects to other Squeak users

Flasher

A red dot that "flashes" on and off.

Frame Rate

A readout that allows you to monitor the frame rate of your system. (**Who can write more what this does?**) From the frame rate's menu, you can choose to change font and its emphasis and you can select pango (**Who can describe this?**).

FreeCell

The objective of FreeCell is to move all of the cards to the four "home cells" in the upper right corner. Each home cell will hold one suit and must be filled sequentially starting with the Ace. There are four "free cells" in the upper left corner that can each hold one card. Cards can be moved from the bottom of a stack to a free cell or to another stack. When moving a card to another stack, it must have a value that is one less than the exposed card and of a different color.

Grab Patch

Use Grab patch to get a partial screen shot. Grab patch is a multi-directional rectangle.

Gradient

A rectangle with a horizontal gradient

Gradient (slanted)

A rectangle with a diagonal gradient

Graph

A graph of numbers, normalized so the full range of values just fits my height. I support a movable cursor that can be dragged with the mouse.

cursor

The current cursor location, wrapped back to the beginning if appropriate

sample at cursor

The sample value at the current cursor location

Graph - sampling

cursor

The current cursor location, wrapped back to the beginning if appropriate

sample at cursor

The sample value at the current cursor location

last value

The last value obtained

clear

Clear the graph of current contents

load sine wave

Load a sine wave as the current graph

load sound

Load the specified sound into the current graph

reverse

Reverse the graph

play

Play the current graph as a sound A magnifying glass that also shows Morphs in the Hand and displays the Hand position.

GStreamerPlayer

A player for OGG movies. It has menu, open, rewind, play, stop, and quit buttons along the top. The menu has the options previous frame, next frame, zoom and advanced (turn on repeat) in addition to rewind, play and stop.

GStreamerPlayer Basic Buttons:

play

Start playing the movie/sound. *stop* Stop playing the movie/sound. *rewind* Rewind the movie/sound.

The object also has soft/loud and start/end sliders below the buttons. It has 13 tiles in the special viewer category movie controls.

- **GStreamPlayer's volume** returns and allows adjustment of the movie/sound volume.
- **GStreamPlayer play** is a command to start playing the movie/sound.
- **GStreamPlayer play until position** is a command to play the movie/sound until the position indicated in the numeric input.
- **GStreamPlayer stop** is a command to stop playing the movie/sound.
- **GStreamPlayer rewind** is a command to rewind the movie/sound.
- **GStreamPlayer's is running** returns true/false if the movie/sound is/is not running.
- **GStreamPlayer's repeat** sets true/false for movie/sound repeating or not. You can also access the state of the movie/sound repeating or not in a watcher.
- **GStreamPlayer's position** is a number representing the current position of the movie/sound.
- **GStreamPlayer's total frames** is the length of this movie in number of frames.
- **GStreamPlayer's total seconds** is the length of this movie in seconds.
- **GStreamPlayer's frame graphic** is a graphic for the current frame.
- **GStreamPlayer's video file name** is the name for the video file.
- **GStreamPlayer's subtitles file name** is the name for the subtitles file.

Image

A non-editable picture. If you use the Paint palette to make a picture, you can edit it afterwards.

Joystick

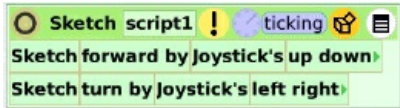


The joystick can be used to control another object.

Drag a joystick into the playfield. Draw an object. Script the object with Forward and Turn.

Open the viewer of the joystick and select "joystick's left right" and "joystick's up down" from the joystick category and put them in place of the numbers besides forward and turn.

You'll have a script that allows the joystick to control the object.



You set the maximum X and Y pixels of "joystick's left right" and "joystick's up down" by using "set X range" and "set Y range" in joystick's menu. This controls the maximum movement of Sketch in the above script. You can also turn "auto-center" off and on and track a real joystick in joysticks menu. Joystick's viewer has the categories joystick and graphics. The graphics category is described in the sketch object section. The joystick category has 6 tiles.

- **joystick's amount** reads the displacement in pixels of the red dot from the center within the joystick box, which is different from the amount Sketch moves in the script above.
- **joystick's angle** reads the angle in degrees from the positive x direction formed by a line joining the center of the box to the red dot.
- **joystick's button1** provides an option to configure for an external joystick button
- **joystick's button2** provides an option to configure for an external joystick button
- **joystick's left right** can be used as an input value as shown in the script above. The reading in a watcher displays the value determined by "set X range"
- **joystick's up down** can be used as an input value as shown in the script above. The reading in a watcher displays the value determined by "set Y range"

Lasso

Use lasso to grab part of the screen with freehand drawing.

Note: you get a picture of the part of screen you grab. A grab of a script or a tool will not be a copy, To make a real copy use the green halo handle

Line

The line behaves the same way as the Arrow object without the arrowhead as default.

Maker Button

This object allow you to make multiple copies of an object and all its scripts and attributes. Simply drag an object onto the Maker Button to load that object into the button. Then each time you click on the button you make a new copy of that object.

Magnifier

A rectangle that you can drag around the screen and "magnify" whatever it covers. Conversely, if the rectangle is stationary, whatever the tip of the mouse points to shows up inside the magnifying glass. From the magnifier's menu, you can choose to change the magnification, track pointer (which turns off and on the second magnifying option described above), show pointer (the center of the magnifier), or change the magnifier to round (*this changes the menu - can someone describe the changes?*).

Magnifier - magnifier

magnification The factor by which the tool magnifies whatever it is looking at *track pointer* Whether or not the pointer should track the cursor location *show pointer* Whether or not the pointer should show the cursor location

Moving Eye

A small black eye with a white pupil. The pupil follows wherever the computer's mouse goes.

MPEGPlayer

A player for MPEG and JPEG movies. It has menu, open, rewind, play, stop, and quit buttons along the top. The menu has the options previous frame, next frame, zoom, subtitles (open subtitles file) and advanced (turn on repeat, set frame rate, create JPEG movie from MPEG, JPEG movie from SqueakMovie, JPEG movie from folder of frames) in addition to rewind, play and stop.

MPEGPlayer Basic Buttons: *play* Start playing the movie/sound. *stop* Stop playing the movie/sound. *rewind* Rewind the movie/sound.

The object also has soft/loud and start/end sliders below the buttons. It has 13 tiles in the special viewer category movie controls.

- **MPEGPlayer's volume** returns and allows adjustment of the movie/sound volume.
- **MPEGPlayer play** is a command to start playing the movie/sound.
- **MPEGPlayer play until position** is a command to play the movie/sound until the position indicated in the numeric input.
- **MPEGPlayer stop** is a command to stop playing the movie/sound.
- **MPEGPlayer rewind** is a command to rewind the movie/sound.
- **MPEGPlayer's is running** returns true/false if the movie/sound is/is not running.
- **MPEGPlayer's repeat** sets true/false for movie/sound repeating or not. You can also access the state of the movie/sound repeating or not in a watcher.
- **MPEGPlayer's position** is a number representing the current position of the movie/sound.
- **MPEGPlayer's total frames** is the length of this movie in number of frames.
- **MPEGPlayer's total seconds** is the length of this movie in seconds.
- **MPEGPlayer's frame graphic** is a graphic for the current frame.
- **MPEGPlayer's video file name** is the name for the video file.
- **MPEGPlayer's subtitles file name** is the name for the subtitles file.

Mines

Mines is a very simple, yet extremely frustrating, game to play. The rules are just this: there are 99 mines laid down on the board. Find them without ""finding"" them. Your first tile is free - click anywhere. The tiles will tell you how many mines are right next to it, including the diagonals. If you uncover the number '2', you know that there are two mines hidden in the adjacent tiles. If you think you have found a mine, you can flag it by either 'shift' clicking, or click with the 'yellow' mouse button. Once you have flagged all of the mines adjacent to a numbered tile, you can click on the tile again to uncover the rest.

NebraskaServer

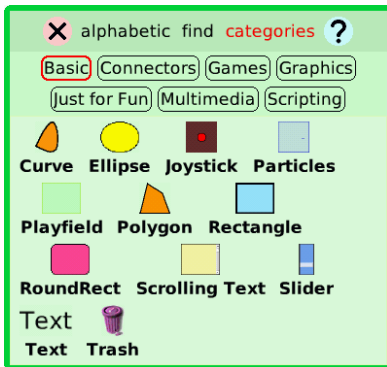
A button to start the Nebraska desktop sharing serverPDA A Personal Digital Assistant

Next Page

A button which, when clicked, takes the reader to the next page of a book. It can be used as an additional page turner in a book. The button can be made opaque or not from its menu. *(I saw no difference between opaque and not - can someone describe the function here?)*

Object Catalog

The "Objects Catalog" allows you to browse through, and obtain copies of, many kinds of objects. You will find the Objects Catalog in the Supplies bin.



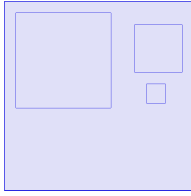
There are three ways to use the Object Catalog, corresponding to the three tabs seen at the top:

- **alphabetic** - gives you separate buttons for a, b, c, etc. Click any button, and you will see icons of all the objects whose names begin with that letter.
- **find** - gives you a type-in pane for a search. Type any letters there, and icons of all the objects whose names match what you have typed will appear in the lower pane.
- **categories** - provides buttons representing categories of related items. Click on any button to see the icons of all the objects in the category.

When the cursor lingers over the icon of any object, you will get balloon help for the item. When you drag an icon from the Objects Catalog, it will result in a new copy of it in your hand; the new object will be deposited wherever you next click.

Note: Not every object belongs to a category, therefore you can find more objects by clicking on *alphabetic* or *find* at the top of the Object Catalog.

Particles



A Kedama World with pre-made components. When you drag out particles, two solid black squares appear. The larger is KedamaWorld and the smaller is patch. Both have special sets of categories and tiles. (*perhaps Yoshiki could describe the tiles or should we refer to the PDF on Kedama?*)

Phone Pad

A device for dialing by tone. You can create the tones by clicking the keys one at a time or enter a key sequence from the menu or from a tile in the special viewer category phonepad which has one tile.

- **PhonePad dial to abc** dials the telephone number in "abc".

Piano Keyboard

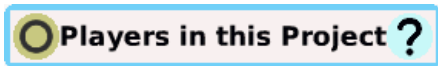
A piano keyboard. You can enable a cord from the menu that allows selection of multiple keys (see below). There are two tiles in the special viewer category keyboard.

- **PianoKeyboard's allowing chord** is true/false depending on whether multiple selection of keys is allowed or not to create a chord.
- **PianoKeyboard's frequency** returns the frequency of the key being played.

Pin

TBD VOLUNTEER NEEDED TO WRITE THIS SECTION

Players



This is a tool that lists all objects in the project. It provides a quick way for the user to access objects in the project by providing a menu for accessibility, a quick button that shows a small thumbnail image of the object, object's viewer, and a tile representation of the object.

Menu

where is this object? - this option reveals the object by surrounding it with its halo.

open viewer for this object - this option opens the viewer for the selected object.

tile for this object - this option creates a tile representation for the object selected.

destroy this object - the object is deleted and put in the trash can when this option is selected. Drag the object out of the trash can to retrieve it.

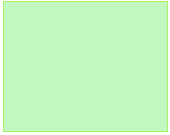
rename object - small text screen pops up when this option is selected. Type the text to replace the name of the object and press "ok".

forcibly rename object - If you want to give this object a name which conflicts with another object's name in the project, use this command. The other object with the same name will in the process be given a different name.

copy object's name to clipboard - As it says, this object name is copied to the clipboard.

inspect object - this launches a Squeak scripting window that shows the variables and codes used to create this object.

Playfield



The Playfield can be thought of as a canvas on which you can place other objects. The Playfield object has certain special properties which allows you to group objects together in a Collection and send messages to all objects contained in the playfield. It is exactly like a holder with three differences which can be seen by in "playfield options..." which is found in the menu: Auto-line layout, Fence enabled (this keeps objects from moving outside of the playfields visible boundaries when using the "forward" tile from the basic or motion categories) and Indicate cursor.

In the viewer it has the special tile categories playfield, pen trails and collections.

auto-line-layout

When checked objects in the playfield are positioned in rows starting at the top of the playfield. When unchecked the objects move back to their original positions.

auto-phrase-expansion

Whether tile phrase dropped onto me should automatically sprout Scriptors around them. By default this is turned off on all playfield's except one, the world's playfield.

automatic viewing

When checked if an object inside the playfield is touched (picked up or moved) a viewer is automatically displayed.

behave like a holder

Sets auto-line-layout, indicate cursor and resize to fit.

Warning: this will change the size and position of all objects in the playfield. Un-checking this option will NOT get you back to your previous playfield state.

fence enabled

This keeps objects from moving outside of the playfields visible boundaries when using the "forward" tile from the basic or motion categories. You can still position objects outside of the playfields visible boundaries by setting the objects x and y values.

grid visible when gridding

Displays the grid if "use gridding" is checked.

indicate cursor

Displays a border around the object at the current cursor position.

mouse-over halos

Displays an objects Halo when you leave the mouse over the object and no other Halo is visible within the your project.

Note: you may need to wait a second for the Halo to appear.

origin-at-center

Set the playfields origin to the center of the playfield. The default origin is the lower left corner of the playfield. The origin can also be set using "set grid spacing".

parts bin

If you drag an object into the playfield a copy of the object is made and placed inside a "Maker Button" This can be used to provide users with playing pieces or building parts for your project.

resize to fit

Resizes the playfield's height to fit the objects contained inside it.

BUG? - Place two items in a playfield of 200 x 200, then place both objects at the bottom left of the playfield, then check "resize to fit" it only adjusts the height. More specifically it only moves up the bottom edge to the lowest object in the playfield. It does NOT adjust the width.

show thumbnails

Whether large objects should be represented by thumbnail miniatures of themselves. (HELP - define large, couldn't figure it out. I put a Rectangle in a playfield and when the rectangle was 100x100 and the playfield was 500x500, then I checked "show thumbnails" it showed a thumbnail of the rectangle. Can someone look at the code)

use gridding

Turns on gridding. When gridding is turned on objects "moved" within the grid will move in "grid spacing increments". For example if you set (using set grid spacing) the width=10 and height=20, the object will not visibly move, until its X is set to the next increment of 10. Ie: if Sketch.X=10 and you increment X by 5, it will take two increments to see X move.

Use "show gridding" to display the grid. Use set grid spacing to set the origin point and the grid size.

round up strays

Brings morphs that are off screen back to the visible screen.

shuffle contents

(see Tiles-Playfield-collection for content, same functionality)

set grid spacing...

Used to set the origin point of the playfield (specified as **x@y**) and the grid spacing (specified as **width@height**)

set thumbnail height...

Sets the height of thumbnail images on the playfield. Note: If you are currently showing thumbnails the new height will not take affect until you uncheck and recheck "show thumbnails"

make detachable

a playfield that has been designated as detachable can have its own local stop-step-go buttons which govern only the status of players within the playfield.

use standard texture

Shows a graph paper background image of 10x10 squares.

make graph paper...

Allows you to specify a grid size, line color and background color for a square grid background image.

show viewers of all players

Shows the viewers of all objects within the playfield's collection that have user written scripts. In this case it does show the viewers of objects within objects in the collection that have written scripts as well.

Only one viewer will be fully visible, the rest will show the icons on the right edge of the World's playfield.

Simply click on one of the icons to expand the viewer. Did you notice that I said the **World's** playfield. Self Study

remove viewers of all players

Hides the viewers of all objects within the playfield.

playfield - Viewer - Category:playfield

graphic

The graphic shown in the background graphic of this object. The graphic is also one of the elements in the Playfield Objects collection.

(NFP (Not For Publication) - At least I believe that was the intended behavior, would think a graphic is more like a background color and should not be part of the collection, but documenting as Etoys 4 works)

(NFP - If you create a playfield graphic then move it with the using the "Move" icon from the Halo, it stays as the Playfield Graphic, but if you "Pick it Up" using the "Pick Up" icon from the Halo it is no longer the Playfields graphic)

mouse x

You get the x coordinate of the mouse pointer relative to the playfield's origin. That means, you get a negative value for x, when you move the mouse to the left of the playfield and a value bigger then the width of the playfield when moving to the right.

mouse y

You get the y coordinate of the mouse pointer relative to the playfield's origin. That means, you get a negative value for y, when you move the mouse below the playfield and a value bigger then the height of the playfield when moving above the playfield.

round up strays

Bring all objects in the playfield, whose x,y position is outside the bounds of the playfield back into view. **Quick Quiz:** So, is it possible to have objects that are in the playfield where you can see at least part of them, but they are "outside the bounds of the playfield"? How can you test your answer?

unhide hidden objects

All objects have a "hide" action that you can run from the "miscellaneous" category. This is telling all objects in the playfield to show themselves.

Quick Quiz: After reading this section is it possible to click on "unhide hidden objects" and still not see all objects? Prove it!

playfield - scripting

tell all contents

Send a message to all the objects contained in the playfield.

If an object receives a message and the message matches the Name of a script, that script will be executed once.

playfield - collections

Certain Objects in Etoys such as Holder and Playfield allow you to group items together in a **collection**. The items in the collection could be of the same type (ex: all text) or different types (sketches, text, polygons, etc). The **collection** allows you to:

- iterate through each object using the cursor
- reference objects in the collection in other scripts
- reference graphics in the collection in other scripts
- reference attributes of the objects in the collection
- Send messages to all objects in the collection with a single Script tile.

Collections allow you to do many things such as:

- Create an animation by changing the graphic of an object by iterating through the graphics in a Holder's collection.
- Store a set of data you have collected and iterate through the items to create a graph, or find the sum, median, average, max or min.
- Send messages to all contents of the collection (see category scripting: "tell all contents") such as reset to the beginning position of a game and/or reset all scores in a game. Simply ensure that each object in the collection has a "reset" script. Then depending upon the object each can have its own specific "reset" behavior.
- Iterate through items to sort them based upon a certain attribute you wish to order by, such as numeric value, vertice count (to sort polygons), for those objects with color attributes by hue, or any other numeric attribute of an object.

cursor

The index of the chosen element

count

How many elements are within me

first element

The first object in my contents

graphic at cursor

the graphic worn by the object at the cursor

include

Add the object to my contents at the end of the collection

include at cursor

Add the object to my contents at my current cursor position

number at cursor

the index number at the cursor

player at cursor

A reference to the object currently at the cursor.

This tile can be put into scripts replacing the scripting tiles object being sent the message. So lets say you have a collection (in a Holder or Playfield) where you store a set of numbers (from some data you collected) and you want to plot those numbers on a graph.

There is a special menu item for this Tile: "Tiles to get". "Tiles to get" lets you reference at tributes of the "Player" (aka object) at the cursor.

remove all

Remove all elements from the playfield. The elements are removed forever.

shuffle contents

Shuffle the contents of the playfield. Note if the Playfield contains a graphic as one of its element, this element is excluded from the shuffling and remains the last element in the collection.

string contents

A concatenation of the the Objects Characters value (in the Basic Tiles Category) of if the Object's Name if the Characters value does not exist.

(Question: why does is use the Characters for Text Objects, but the Object Name for everything else?)

tell all contents

Send a message to all the objects contained in the playfield.

If an object recieves a message and the message matches the Name of a script, that script will executed once. NOTE: It does not send messages to objects contained in objects in the playfield.

For example:

- If you have a *Sketch* inside a *Holder* which is inside a *Playfield*
- Both the *Sketch* and the *Holder* have scripts named "script1"
- When the *Playfield* sends the message "script1" to all contents, only the *Holder's* "script1" will execute.

pen trails

These tiles have a different behavior from those in the "pen use" category. These tiles refer to the pen trails of the objects contained in the playfield. Where "pen use" refers to pen trails made by the playfield.

The Collections and Menu Items for "pen trails" are the same but different names.

batch pen trails

MenuName: "batch pen trails"

Whether pen trails should reflect small movements within the same tick or only should integrate all movement between ticks

clear pen trails

MenuName: "clear pen trails"

Clears the pen trails of objects in the collection.

Does not clear the pen trails of objects within the objects in this collection or of the playfield itself.

has pen trails

This is a Boolean to indicate whether there are any pen trails in the playfield. It does not check for pen trails inside objects in the collection.

lift all pens

MenuName: "all pens up" Tells all objects in the playfield to set the "pen down" value to false for all objects contained in the playfield.

lower all pens

MenuName: "all pens down" Tells all objects in the playfield to set the "pen down" value to true for all objects contained in the playfield.

pen trail graphic

This is a graphic of all pen trails inside the playfield. It could be used if you had multiple objects drawing pen trails inside the playfield to make a picture. Then you could set the graphic of another object to the picture of the combined pen trails.

trail style for all pens

Tells all objects in the playfield's collection to set their pen style to the value specified

- lines (MenuName "all pens show lines")
- arrowheads (MenuName "all pens show arrowheads")
- arrows (MenuName "all pens show arrows")
- dots (MenuName "all pens show dots")

playfield - Halo

initiate painting (Include Painting Image)

I am flagging "initiate painting" for removal. Reason as per Chapter Talk in Tiles Chapter this feature is flagged for removal) (NOTE: not documenting Unlock "NameOfGraphics ketchUsedAsBackground". This only shows up in the Playfield's menu, if you have a background graphic, see initiate painting above, which is one way to create a background graphic) Rita: I don't see where initiate painting has anything to do with halo, so it should be removed from here.

playfield - sound

sound pitch pitch of sound *sound level* level of sound

dial number dial number of sound

listening whether the stethoscope is listening

playfield - input

last keystroke The last unhandled keystroke

playfield - preferences

use vector vocabulary Whether to use the Vector vocabulary with etoy scripting in this project

drop produces watcher Whether a drop of a value tile, such as "car's x", on the desktop, should produce a watcher for that value

allow etoy user custom events Whether to allow "custom events" in etoys.

batch pen trails Whether pen trails should reflect small movements within the same tick or only should integrate all movement between ticks *fence enabled* Whether an object hitting the edge of the screen should be kept "fenced in", rather than being allowed to escape and disappear

keep ticking while painting Whether scripts should continue to run while you're using the painting system

olive handle for scripted objects Whether the default green halo handle (at the top right of the halo) should, for scripted objects, be the olive-green handle, signifying that use will result in a sibling instance.

implicit self in tiles Whether tiles representing a player should be suppressed in Viewers and Scriptors belonging to that player

Polygon



Polygons are graphical objects with defined corners or *vertices* that can be grabbed and dragged to change the shape and size of the object. Although the technical definition of a polygon is a closed plane figure with straight edges, polygons in Etoys may be open with curved edges.

A new polygon has four vertices and four sides.

Select "Show handles" from the Halo Menu to display circles at each of the vertices. You can also click on the polygon holding the shift key down to bring up the handles. Grab the circles to adjust the lengths of the sides of the polygon. "Show handles" will also display triangles at the midpoints of each side. Grab a triangle to "split" the side and create a new vertex at the triangle.

The number of vertices may also be controlled via the polygon pane of the Viewer ("vertices count"). Vertices may only be removed via the Viewer or via a script.

An Etoys polygon with two vertices looks like a line segment. An Etoys polygon with one vertex looks like a point (and is very hard to click!). An Etoys polygon may not have less than one vertex.

Each vertex of the polygon is assigned a cursor number. Reference a vertex by assigning the vertex's cursor number to the polygon's "vertex cursor" property on the polygon pane of the Viewer. The currently-referenced vertex appears as a blue circle if the polygon's handles are visible. The "x at cursor" and "y at cursor" properties reference the selected vertex's location.

The polygon pane of the properties view also provides access to the following Etoys polygon properties:

- line is curved (Boolean, controls if sides of the polygon are straight or curved).
- line is opened (Boolean, controls if the vertex with the highest cursor number is connected to the vertex with cursor number = 1). This creates an open figure.
- showing handles (Boolean).

The polygon pane of the Viewer also contains scripting tiles to add a vertex at the beginning (creating a new vertex at cursor number = 1, at the current cursor position, and at the end. Note that new vertices have the same location as the vertex at the currently selected cursor position. Creating new vertices will not appear to change the shape of the polygon until the location of the new vertex changes.

The polygon pane of the Viewer contains a scripting tile to remove the vertex at the current cursor position.

The polygon pane of the Viewer also contains a scripting tile to "shuffle vertices". This action shuffles the cursor position of the vertices in the list of vertices (not the x@y location of the vertices). As a result, the shape of the polygon may change, as the vertices appear to be connected in different ways.

The angle at a vertex may be calculated by determining the lengths of the sides of a triangle formed by the vertex and the endpoints of the segment sides, and using the Law of Cosines. Although Etoys does not have an arccos function, it does have an arctan function. The following identity may be used to find the arccos from the arctan:

$$\text{ArcCos}(x) = \text{ArcTan}(\text{Sqrt}(1 - \text{Sqr}(x)) / x)$$

Unique Tiles

vertex cursor - The index of the chosen vertex of vertices

vertices count - How many vertices are within me

x at cursor - The x coordinate value at the vertex cursor

y at cursor - The y coordinate value at the vertex cursor

add a vertex at beginning - Add a vertex at the beginning of my vertices list

insert a vertex at cursor - Add a vertex at my current cursor position

add a vertex at end - Add a vertex to the end of my vertices list

remove all vertices but cursor - Remove all vertices but my current cursor position vertex

remove the vertex at cursor - Remove the vertex at my current cursor position

shuffle vertices Shuffle the vertices

line is curved Whether the line is curved

line is opened Whether the line is opened

showing handles Whether the handles are showing

Preferences

Allows you to control numerous options.

Previous Page

A button which, when clicked, takes the reader to the previous page of a book. It can be used as an additional page turner in a book. The button can be made opaque or not from its menu. **(I saw no difference between opaque and not - can someone describe the function here?)**

ProjectHistory

A tool that lets you navigate back to recently-visited projects

Random

This creates the tile that generates a random number in a given range. This functionality is also available in the Scripts supplies bin amongst other tiles that can be used for scripting.

Random Connectors



A button that will create random arrows each time you drag one from it.

Rectangle



This is a basic rectangle. The Rectangle object is not related to the Polygon object and does not provide important options available with Polygon objects. Rectangle has a color category of tiles that allows you to use its red, green, blue, alpha, hue, brightness, saturation properties in programs.

Round Glass

Similar to the Magnifier, except that it is round instead of rectangular. **(Can someone describe the menu options for this object?)**

RoundRect



A rectangle with rounded corners. It has the same color category as rectangle.

Ruler

A rectangle which continuously reports its size in pixels

Same

The object of SameGame is to maximize your score by removing tiles from the board. Tiles are selected and removed by clicking on a tile that has at least one adjacent tile of the same color (where adjacent is defined as up, down, left, or right). The first click selects a group of adjacent tiles, a second click in that group will remove it from the board, sliding tiles down and right to fill the space of the removed group.

Schematic Connectors



Schematic connectors differ from connectors in two ways, one they can not have arrow heads and two as you move the objects around the connection points change.

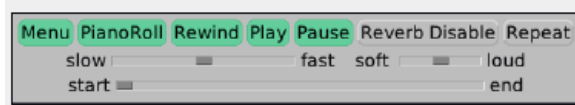
Not sure of the need for this or why moving connection points adds value.

Scripting

This creates a small space for confined scripting. The space includes a small playfield, a trash can and a dedicated *All scripts* type player. The player will be limited to the scripts within the *Scripting* area. This scripting area as well as the playfield within this space can be enlarged using the resizing tool on their respective halos.

ScorePlayer

The ScorePlayer object plays MIDI music scores. MIDI stands for "Musical Instrument Digital Interface", a standard format for digitally representing musical scores.



Click the Menu button to display a menu of ScorePlayer options. Click the "open a MIDI file" menu option to load a MIDI score. Etoys will display all MIDI files in the Etoys folder. You can also save as AIFF file, save as WAV file, save as Sun AU file, reload instruments, play via MIDI, and make a pause marker from the menu. Once a MIDI file is loaded, the instruments referenced in the MIDI score will appear as part of the ScorePlayer object. >



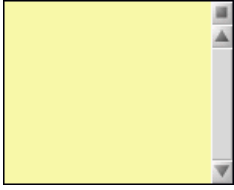
Click on an instrument name to select a different instrument.

Click the PianoRoll button to display a PianoRollScore object representing the loaded MIDI file. If no MIDI file is loaded, the PianoRollScore will be blank.

Click the Play button to listen to the MIDI score.

<

Scrolling Text



A scrollable, editable body of text. From the menu, you can open and close editing (*does anyone know what this means?*), make the scroll bar retractable or inboard, and put the scroll bar on the left or right. It has the same color category as rectangle.

Slider



A scriptable control that allows you to choose a numeric value by dragging a knob. It has the same color category as rectangle. From the menu you can open and close editing (*does anyone know what this means?*), set action selector (*does anyone know what this means?*), change arguments (*does anyone know what this means?*), set minimum value (see below), set maximum value (see below), switch between number values ascending and descending as you move the slider up and down (see below), and turn on and off truncation (see below). Slider's viewer has the extra category slider with 9 tiles.

- **slider's width** sets the width of the slider. This can also be set using the yellow change size button on the slider's halo.
- **slider's color** changes the color of the slider's background.
- **slider's descending** determines whether the max val is at the top or bottom of the slider. This can also be set on slider's menu.
- **slider's height** sets the height of the slider. This can also be set using the yellow change size button on the slider's halo.
- **slider's knob color** changes the color of the slider's knob.
- **slider's max val** sets the value at the top of the slider. This can also be set on slider's menu.
- **slider's min val** sets the value at the bottom of the slider. This can also be set on slider's menu. By choosing a negative value, you place zero somewhere between the top and bottom.
- **slider's numerical value** is the value of the slider position measured from zero. The value read and position of zero on the slider depend on the numbers you input to sliders max val and min val.
- **slider's truncate** is a true/false value that can also be set in the slider's menu. To see its intended purpose set the Max Value to 10, Drag out a watcher for slider's numeric value and move the slider. With truncation=false, the numeric value acts like a float, With truncation set to true it will only return integer values.

Note that you can change the slider to horizontal by using the yellow change size button on the slider's halo or the slider's width and height tiles. When the width is greater than the height, the slider is horizontal and then up and down go to left and right in the above descriptions.

Sound Library

This tool allows you to view and manage the "Sound Library", which is the list of named sounds that can be used in the tile-scripting system.

Click on a sound name in the list to select it. The buttons at the top of the tool apply to the sound you have selected.

Play button -- press this to start playing the selected sound.

Stop button -- if the selected sound is playing, pressing this will stop it.

Tile button -- Click on this to obtain a scripting tile representing the selected sound.

Rename button -- allows you to rename the selected sound.

Delete button -- allows you to delete the selected sound from the Sound Library. All tiles that formerly pointed to this sound will be changed to point to "croak" instead.

Load button -- allows you to load a sound into the Sound Library from a file.

You can also add sounds to the Sound library using the Sound Recorder, and also by dragging an external sound file (e.g. a file with extensions .wav or .aif) into Etoys.

Note: the "universal" sounds built in to the system cannot be renamed or deleted.

Additionally, a command for opening a "wave editor" tool on the selected sound can be found in the tool's halo menu.

Sound Library has a question mark button at the upper left with excellent help for using this object. The above text is from that help section.

Sound Recorder

Basic tool to record sound. *Note : You must have microphone input for this tool to record.*

Press "Record" to start recording. Press Stop when finished recording.

After making a recording, you can:

Press "Play" to play back the recording.

Press "Record" to start a new recording
(the old one would be discarded).

Press "Save" to save the recording in the sound library.

Press the menu icon to get a menu with further options. The menu icon:

1. Shows the duration of the recording
2. Allows you to open a help flap which displays the information above and below.
3. Hands you a sound token (see below).
4. Let's you choose from the following compressions:
 - Speex (for speech)
 - Vorbis (for music)
 - GSM (basic noise inducing compression)
 - No compression, Note: sound can be huge without compression!
5. Allows you to open the sound in wave editor (see description below).

If you wish to refer to the sound in scripts, you need to add it to the sound library; press Save to do that; you will need to supply a name for it.

If you want to retain the sound but do not need to refer to it in scripts, you need not name it; instead, use "hand me a sound token", found in the menu, to obtain a little "sound token" object that you can subsequently use in a variety of ways:

- You can double-click on the sound token to hear the sound again.
- You can decide to save the sound after all, by using an item in the sound token's halo menu.
- You can drop the sound token into a PianoRoll or an EventRoll.

A device for analyzing sound input. It has buttons along the top to start and stop the analyzer. When the analyzer is running, the "on" button at the right turns yellow. The menu button allows the user to set the sampling rate, FFT size and select the display type (signal, spectrum and sonogram).

Star

Star can be useful when simulating gears.

You can change the number of points via the halo menu or by shift clicking on the star.

Shift clicking brings up a novel tool at the center of the star. Green triangle up adds more points to the star, pink down reduces the number of points. Green right shortens the length of the points, pink left give you longer points.

vertices count How many vertices are within me *star ratio*

The menu options are

- more sides
- fewer sides
- twinkle fatter
- twinkle thinner

Star has its own Viewer category where you can set the number of points and their length with tiles. It has the same color category as rectangle.

Status

Buttons to run, stop, or single-step scripts

Sticky Pad

A translucent, borderless rectangle of a standard size, delivered in a predictable sequence of pastel colors each time you drag one out from the object catalog

Storyboard

A storyboard authoring tool

Tetris

Tetris, yes Tetris

Text

Text

Text can be used just as for any other program. It can be used for stories, labels, or other purposes. Drag the text into the playfield and change the font, size, color, and style by right clicking on the text and using the appropriate handles at the bottom of the halo. From the menu, you can turn text autofit off and on, wrap text to bounds, make the text translatable, and use pango (*Can someone describe how using pango affects layout, rendering and internationalization, IE: Why would someone use this*). The menu also provides text formatting options like text properties (including some not available through the bottom handles like text color), text margins, add predecessor (*does anyone know what this means?*), add successor (*does anyone know what this means?*), code pane menu, code pane shift menu, and fill owner's shape. The menu also allows you to make a holder for the characters. It has the same color category as rectangle. Text's viewer has the extra category text with ten tiles.

- **text's all but first** returns all the text but the first character in the text object.
- **text's character at cursor** returns the character at the cursor which can be set by **text's cursor** (see below).
- **text's characters** returns all the text in the text object.
- **text's count** returns the number of characters in the text object.
- **text's cursor** the position of the cursor in the text object.

- **text's first character** returns the first character in the text object.
- **text's insert characters abc** inserts the character appearing in the abc section of the tile at the current cursor position. The characters in **abc** can be edited by clicking on **abc** and typing new characters.
- **text's insert contents of dot** inserts the contents of dot. (*Has someone explained how dot works somewhere? It should be referenced here. - This should be hidden see SQ-812 for explanation of what it does*)
- **text's last character** returns the last character in the text object.
- **text's numeric value** the numeric value of the text string, if all the text will be replaced by a string representing the numeric value

additional tiles in color category:

color - The color of the text

background color - The color of the background behind the text

Text (border)

A text field with border

Text chat

A tool for sending messages to other Squeak users

Text chat+

A tool for sending messages to several Squeak users at once

Text Ellipse



Creates an ellipse with a Enhanced text box inside it. Type your text inside the box, which resizes to fit the words

Text Rectangle



Similar to a text ellipse except this is a rectangle. Type your text inside the box. It will re size to accommodate the words you type.

Thread Navigator

A tool that lets you navigate through a thread of projects. When you drag out the thread navigator, it immediately attaches itself to the lower right. Clicking the orange circle at the center gives a list of commands to choose from to navigate through projects. Orange next and previous arrows appear at the right and left to allow navigation. For example, you can create a thread of all projects and a holder appears with all the projects currently available. You can drag the projects to change the order and when you click the "okay" button, you are asked to supply a name for the thread navigator.

Title



Enhanced Text that is pre-formatted to be large and bold.

Trash



A tool for discarding objects. For convenient access to the trash can, drag it onto your play field. Then any object can be dragged into the trash easily. To remove an object, drop it on any trash can. To view, and maybe retrieve, items that have been thrown away, double-click on any trash-can. Things are retained in the trash-can if the "preserveTrash" preference is set, otherwise they are purged immediately. From the menu you can choose to make the trash can opaque or solid. Trash's viewer has the extra category trash with two tiles.

- **trash's empty trash can** empties the trash can if the "preserveTrash" preference is set.
- **trash's page count** returns a number of pages in the trash can. Every object you put in the trash can goes into a new page.

Triangle

A three-sided polygon. One can also create this object using the *Polygon* object. It shares much functionality and tiles as the *Polygon* but is limited to three vertices by default. It is also different from a *Polygon* in that if you "shift-click" on it, it doesn't have the option of adding vertices using the green triangle handle. However, one can create a polygon with more than three sides by adding vertices by using the "add vertex in the begging (or end)" tiles and dragging these vertices out to form another side to the triangle. {Please see the *Polygon* object in the ObjectsCatalogBasics chapter for the shared functionality and descriptions with this object}

TrueType banner

A short text in a beautiful font. Use the resize handle to change size.

Wave Editor

A workbench for seeing and editing wave forms. It has the buttons menu, play, play before, play after, play loop, test, save, set loop end, set one cycle, and set loop start across the top. Using the menu attached to a sound recorded in Sound Recorder object (described above), you can open the sound in the Wave Editor object. *(Is there another way to open a sound on Wave Editor. Can someone supply the details?)*

Welcome

A sign that you accept morphs dropped directly into your world

World

The *world* is the main project page you see when you open an Etoys project. It is basically a playfield (see chapter 5.x, playfields). Because of that the *world* can contain individual objects within it, such as drawings, but can also contain more complicated objects such as *books*, which tend to be made up of many different objects.

World Buttons

If you click with the right mouse button on the *world*, you'll see the *world's* halo, which is simpler than other halos, containing only five buttons.



The pink eyedropper which allows you to change the world's color by picking a color

from anywhere on the screen.



The gray pencil allows you to draw directly onto the world by bringing up the painting tool and using the whole screen as a canvas. Every object you draw directly on the world will be embedded in the world and will be locked (which makes it inert to all user interactions). To change that behavior, repeatedly right click on the object and use the objects menu (see chapter ... for more information about

objects menus).



The turquoise eye brings up the viewer for the world. Note that the world's viewer has a simpler set of tiles within it that don't give you all the options of

other types of objects. (See chapter 3 for more information on tiles.)



The orange icon will give you a tile representing the world. This way you can add a reference to the world

in a script.



The click on the menu icon gives you several options. **World's options**

about this system - This option shows you what version of Etoys you are running and allows you to open the license information.

redraw screen - This option will redraw your entire screen.

preferences - This entry opens the preference pane, where you can change the default preferences. You can find out more about the available preferences in chapter ...

authoring tools - This menu option allows you to view authoring tools that control the world. These include options such as "remove all viewers" and "etoy vocabulary summary" as well as "attempt misc repairs."

display mode - This menu option allows you to scale your project to fit screens other than XO laptop screens.

desktop color - This menu option lets you change how the world's background looks, and includes options such as gradient fill, and bitmap fill.

pen trails - This menu option lets you control how the pen trails from all objects (but not the objects within those objects) within the World behave. You can, for example, make all pen trails arrowheads, make all pens up, clear all pen trails, etc.

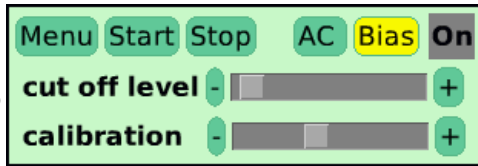
playfield options - This menu option opens the playfield menu for the world. You can find more information about playfield menu in chapter ...

World Stethoscope

With the WorldStethoscope object, tiles use sound signals from the computer's microphone input to create programs. WorldStethoscope Etoys tiles read the frequency and level of the microphone input. The WorldStethoscope object is brought into the world from the ObjectCatalog multimedia category by grabbing the WorldStethoscope object and dragging it out onto the world. Click the Start button on the WorldStethoscope object to begin receiving the signal from the microphone input. The On button will turn yellow. The Menu button at the left allows the user to set the sampling rate, set the FFT size, show adjustments and add a graph. Show adjustments allows the user to show and hide sliders for adjusting the cut off level and calibration. The "add a graph" choices show a sonogram, the signal, the frequency spectrum or data. The stop button discontinues the signal acquisition. The AC and Bias buttons were designed specifically for the XO computer, which allows for AC or DC signal input.

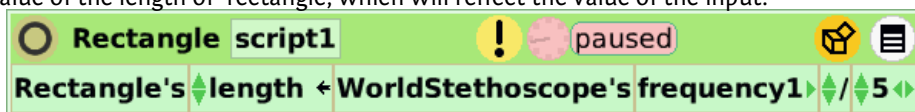


To access tiles that allow you to develop programs using the input signals, open a viewer for the WorldStethoscope object from its halo and choose the Sound category. The WorldStethoscope's sound category has 13 special tiles not available to other objects.



- **WorldStethoscope's cut off level** filters out sounds with levels greater than this value. **WorldStethoscope's cut off level** 10000
- **WorldStethoscope's dial number** is a DTMF (Dual Tone Multi Frequency) decoder. This value reflects a decoded dial tone number. This function is useful for using a cell phone as a remote controller. Pull out the watcher for this tile with WorldStethoscope on, drag out a PhonePad object from the Multimedia category of Object Catalog, and click a key on the PhonePad. The watcher shows the clicked key. **WorldStethoscope's dial number** 0
- **WorldStethoscope's frequency1** returns the frequency of the highest peak in the FFT spectrum. **WorldStethoscope's frequency1** 0
- **WorldStethoscope's frequency2** returns the frequency of the second highest peak in the FFT spectrum.
- **WorldStethoscope's frequency3** returns the frequency of the third highest peak in the FFT spectrum.
- **WorldStethoscope's level1** returns the input level of the highest peak in the FFT spectrum.
- **WorldStethoscope's level2** returns the input level of the second highest peak in the FFT spectrum.
- **WorldStethoscope's level3** returns the input level of the third highest peak in the FFT spectrum.
- **WorldStethoscope's listening** returns true/false when WorldStethoscope is/is not working.
- **WorldStethoscope's pass over** filters out sounds with frequencies less than this value.
- **WorldStethoscope's pass under** filters out sounds with frequencies greater than this value.
- **WorldStethoscope's read out** is for DC signals input on the XO computer. The "bias", or the internal voltage from the plug can drive the sensor plugged into the audio plug. When the sensor changes resistance, **read out** is the (digitized) raw value from the sensor. (IIRC, 0V to 5V outputs are mapped to 0 to 32767.)
- **WorldStethoscope's update interval** specifies how often the input is read.

Note: You cannot use multiple WorldStethoscopes and/or SoundRecorders at the same time. If you want to use the frequency1 tile to build a meter, the tile can be dropped onto the value of the length of rectangle, which will reflect the value of the input.



The WorldStethoscope was developed in Japan by Kazuhiro Abe and Tetsuya Hayashi. Hardware developed by Kazuhiro Abe and Tetsuya Hayashi turns a voltage signal into a frequency so any circuit element that outputs a voltage can be turned into a sensor. For example, thermistors can be used to measure temperature, light sensitive diodes can be used to measure light levels and variable resistors can be used for control signals. Using the WorldStethoscope tiles young learners can develop an interface for data acquisition, control and display. The learner has more ownership and can develop a deeper understanding by creating the interface as opposed to simply using a canned data acquisition program. It could be used in creating a meter to display the frequency reading or to control a process based on the frequency reading. Depending on the element connected to the input, the frequency may be a measure of temperature, light level, resistance, location, or many other things. By calibration of the output of WorldStethoscope the reading can be converted to an actual temperature, light level, etc. Perhaps the easiest thing to do with WorldStethoscope is to attach a microphone to the computer input and measure the frequency and level of the sound input.

The "add a graph" choices on the menu button are most useful for a sound signal input. If you choose "data", "spectrum" or "signal", you get an object with the following additional tiles:

data color The color hue of the data

sampling

cursor The current cursor location, wrapped back to the beginning if appropriate

sample at cursor The sample value at the current cursor location on>

last value The last value obtained

Clear the graph of current contents

relative scale Whether the display scale is relative to the actual maximum and minimum values in data

max val The maximum of the range drawn in the graph

min val The minimum of the range drawn in the graph. If you choose "sonogram", you get a Sonogram object with the following additional tiles:

data color , color The color hue of the data

sampling scroll delta The horizontal scrolling stride when data is going off the edge

As an example of a more complex classroom project, students could make their own position sensor by laying a series of nearly connected wires along a track and as a metallic ball rolls across the nearly connected wires the circuit would be completed and a signal would be sent to the WorldStethoscope Etoys project. The signal could be monitored as a function of time and the speed of the ball could be determined. Although there are many canned devices available to accomplish this goal, the process of creating sensors and designing an interface to analyze and display the results is far more valuable than merely taking the data.

Workspace:

A Workspace is a simple window for editing text. You can later save the contents to a file if you desire.

Zip Tool

A viewer and editor for Zip archive files

APPENDICES

License

All chapters copyright of the authors (see below). Unless otherwise stated all chapters in this manual licensed with **GNU General Public License version 2**

This documentation is free documentation; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this documentation; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Authors

AppendixEtoysFriendlyOff

© Rita Freudenberg 2009, 2010

AppendixKnownErrors

© Rita Freudenberg 2010

AppendixMIMETypes

© Rita Freudenberg 2009, 2010

AppendixMorph

© Rita Freudenberg 2009, 2010

Credits

© adam hyde 2006, 2007

GETTING STARTED

© Anne Gentle 2009

Modifications:

Bert Freudenberg 2009

Cherry Withers 2010

ChristineS Murakami 2009, 2010

John Curwood 2009

Randall Caton 2010

Rita Freudenberg 2009, 2010

INTRODUCTION

© adam hyde 2006, 2007

Modifications:

Anne Gentle 2009

Bert Freudenberg 2009

ChristineS Murakami 2009

John Rigdon 2010

Randall Caton 2010

Rita Freudenberg 2009, 2010

Stephen Thomas 2009, 2010

OBJECTS

© Anne Gentle 2009

Modifications:

Bert Freudenberg 2009

Cherry Withers 2009, 2010

ChristineS Murakami 2009

Jim Davis 2009

Karl Ramberg 2009
Randall Caton 2009
Rita Freudenberg 2009, 2010
Stephen Thomas 2009, 2010

COMMONTILES

© Anne Gentle 2009
Modifications:
Bert Freudenberg 2009, 2010
Randall Caton 2010
Rita Freudenberg 2009, 2010
Stephen Thomas 2009, 2010

USER INTERFACE

© Anne Gentle 2009
Modifications:
Bert Freudenberg 2009
Cherry Withers 2010
ChristineS Murakami 2009
Karl Ramberg 2009
Rita Freudenberg 2009, 2010
Stephen Thomas 2010



Free manuals for free software

General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you

must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a)** Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b)** Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c)** Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

*License: The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, **you agree** that your edits can be used under both the **GPL and MIT** licenses.*

Appendix I: Etoys Friendly Off

More options, tiles and menus you get when you turn Etoys-Friendly off

Playfield

These are tiles you get in playfields viewer category collections.

include at end(NOT in Etoys Friendly Mode)

Add the object to the end of my contents list.

include at beginning(NOT in Etoys Friendly Mode)

Add the object at the beginning of my contents list.

viewer category display

show navigation bar

Show the navigation bar at the top of the screen

hide navigation bar

Hide the navigation bar at the top of the screen

use blueprint canvas

Display the world as a blueprint

use normal canvas

Display the world normally

use blueprint canvas

Display the world as a blueprint

use normal canvas

Display the world normally

Appendix: Morph

License: The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, you agree that your edits can be used under both the GPL and MIT licenses.

About Siblings (by Karl)

If you hold down shift while copying with the green halo handle you make a copy.

Almost... What you now have are two **sibling players**, each of which is a "Player" or "Object" in its own right. (Also copies made by the Player Copy tile are siblings.) Each player wears its own separate Morph as its costume. The siblings are related in that they share certain things in common, in particular: scripts and variable definitions. So if you add a variable to one sibling, all siblings will have their own instance of that variable with the same name. The same name, but the values can be different. For example if I add a variable called "speed" to one sibling, all siblings will have a variable with that name. But I can have one sibling with speed=5 and another with speed=10. Same variable name different values.

Also if I add a script to one sibling, all siblings will have the same script (and same scripting tiles, if I change it in any sibling at anytime). But although siblings share a script, each sibling has its own private "status" for that script, i.e. remembers separately whether the script **as used by this instance** is "normal", "paused,", or "ticking", or set to trigger on mouse-up, etc.; and what the tick-rate is.

When using siblings, what's really happening underneath is that there is a custom Player "Class" is created to bear the shared code, and the siblings are individual "Instances" of that class. "Class" and "Instance" are among the most fundamental concepts of object-oriented programming

So why would you use siblings? Well besides teaching one of the most fundamental concepts of object oriented programming, you could use siblings in a game to make characters that have similar behaviors (aka scripts) but different looks (you can change the looks by either re-drawing the sketch or by setting the Sketch's graphic (if the object is a Sketch or image you copied in) or changing its size and colors. You could also have ships in a game that keep track of how many times they were hit (using a variable to count the # of hits). That way you can program a basic ship once and make multiple siblings. Then if you decide to change the behavior (ex: modify a script, or add a new variable), you can change it once and all the siblings are changed.

Siblings could also be used to try to simulate or model real world behavior. For example deer population or some other animal. You could create an Object that simulates the behavior of a deer: searches for food, ages, breeds (at a certain age), etc. and make sibling copies of that one object. Then each Sibling (aka individual instance of the Class Deer which you have defined) could age, search for food (and find it or not which could affect its life expectancy), etc at its own rate. You could also create predators and food that grows. Then run multiple simulations varying the amount of food, starting number of deer, # of predators, etc.

#NOTE: Need to point to some good examples in Etoys and lesson plans from Internet

If the "all instances" box is not checked in AllPlayers, you'll see the scripts for only one player of each sibling group. This will give you "control" over the scripts of only one of the players. If you want to be able to "control" scripts for both of the siblings, make sure the "all instances" checkbox is checked.

Unless the "all instances" checkbox is checked, creation of new siblings will typically not result in any noticeable change in the all- scripts tool, since all the scripts of the new sibling will already be represented by another instance in the tool. Perhaps that's what you're seeing.

Also, if the "tickers only" box is checked, only scripts whose status is "ticking" or "paused" are shown in the all-scripts tool. This can be another reason why the all-scripts tool may

seem not to be "picking up" on a change.

#DEVELOPER NOTE: *We refer to "siblings" but don't have a formal term for the "class". We don't much use the terms "class" and "instance" in the etoys UI, nor in any documentation I'm aware of. (A notorious exception is the "all instances" checkbox in the all-scripts tool.) Hmm... maybe the checkbox should read "all siblings" rather than "all instances"...*

#DEVELOPER REQUEST: *AFAIK, it's fine to have an all-scripts tool open as you create, delete, re name, and otherwise manipulate objects and scripts; I'm not aware of situations where it "misses" anything, but if you can describe a sequence of steps leading to such a situation, it would be helpful.*

License: *The Etoys Manual will be dual-licensed under GPL (standard for FLOSS Manuals) and MIT (standard for Etoys). By contributing, **you agree** that your edits can be used under both the **GPL and MIT** licenses.*

File formats supported by Etoys

A list of all file types that can be opened, displayed, etc. in Etoys. Make a list of pictures, audio, movies, misc.

SWF

Shockwave Flash Files up to version 3 can be opened and played from a file list. A Flash Player is opened.

MIME types for Etoys files

project

Made with Booki

Visit <http://software.booki.cc>