

Shoes Tutorial Note

- For the Shoes App Rookie Creators -

Mar. 13th, 2009 by ashbb (Satoshi Asakawa), citizen428 (Michael Kohl), kotp (Victor H. Goff III)

Table of contents

1. [00100 Introduction](#)
2. [00200 Download Shoes](#)
3. [00300 First step](#)
4. Birds-eye view (Survey basic features)
 - [00401 Concept](#)
 - [00402 No.1 para \(sample1.rb, sample2.rb\)](#)
 - [00403 No.2&3 stack and flow \(sample3.rb, sample4.rb\)](#)
 - [00404 No.4 button \(sample5.rb\)](#)
 - [00405 No.5 image \(sample6.rb\)](#)
 - [00406 No.6 edit line \(sample7.rb, sample7-1.rb\)](#)
 - [00407 No.7 link \(sample8.rb\)](#)
 - [00408 No.8 background \(sample9.rb, sample10.rb, sample57.rb\)](#)
 - [00409 No.9 Shoes.url \(sample11.rb\)](#)
 - [00410 No.10 clear \(sample12.rb, sample13.rb\)](#)
5. Tips for creating our original Shoes apps
 - [00501 Open Shoes built-in manual and Shoes console window](#)
 - [00502 Output messages on the Shoes console window \(sample15.rb\)](#)
 - [00503 shoes --help](#)
 - [00504 App object and coding style \(sample16.rb, sample17.rb, sample18.rb, sample47.rb\)](#)
 - [00505 Built-in Constants and methods](#)
 - [00506 Scope: A tip of using the YAML file \(sample19.rb, sample19-1.rb\)](#)
 - [00507 keypress, mouse and clipboard \(sample20.rb, sample21.rb, sample65.rb\)](#)
 - [00508 The Widget class \(sample22.rb, sample49.rb\)](#)
 - [00509 shape \(sample23.rb\)](#)
 - [00510 mask \(sample24.rb\)](#)
 - [00511 Drawing directly on to images \(sample25.rb\)](#)
 - [00512 Style \(sample26.rb, sample56.rb\)](#)
 - [00513 Shoes.setup \(sample27.rb, sample50.rb\)](#)
 - [00514 Downloader \(sample28.rb\)](#)
 - [00515 Assign Shoes URL dynamically \(sample29.rb\)](#)
 - [00516 Classes List and Colors List \(sample30.rb, sample30-1.rb, sample30-2.rb\)](#)
 - [00517 start, stop and restart \(sample31.rb\)](#)
 - [00518 Combination of image objects show/hide and mouse hover/leave \(sample32.rb, sample33.rb, sample34.rb\)](#)
 - [00519 arc and cap \(sample35.rb\)](#)
 - [00520 widget with block \(sample36.rb\)](#)
 - [00521 text message slide-in \(sample37.rb\)](#)
 - [00522 #! shoes \(sample38.rb, sample38-1.rb\)](#)
 - [00523 loading widgets from other files? \(sample39.rb, sample39-creature.rb\)](#)
 - [00524 optional arguments \(sample40.rb, sample40-1.rb\)](#)
 - [00525 slot with scrollbar \(sample41.rb\)](#)
 - [00526 The :state style \(sample42.rb\)](#)
 - [00527 Shoes::FONTS and External Fonts \(sample43.rb\)](#)

- [00528 Shoes Tutorial Note Launcher \(sample44.rb\)](#)
- [00529 UTF-8 \(sample45.rb\)](#)
- [00530 Open a new app window \(sample46.rb, sample48.rb\)](#)
- [00531 Open the Shoes console window from your app \(sample51.rb, sample55.rb\)](#)
- [00532 Customize Shoes Class \(sample53.rb\)](#)
- [00533 Image Effects with blur method \(sample54.rb\)](#)
- [00534 Video playback \(sample59.rb, sample59-1.rb\)](#)
- [00535 Scope: local variable and instance variable \(sample60.rb\)](#)
- [00536 edit line with block \(sample63.rb\)](#)
- [00537 One way of layer manipulation \(sample64.rb\)](#)
- [00538 Show and hide the slots \(sample66.rb\)](#)
- [00539 class definition outside of Shoes.app block \(sample69.rb, sample69-1.rb, sample69-2.rb\)](#)

6. Hot Topics in the Shoes ML and Shoooes.net

- [00601 External Fonts](#)
- [00602 Locking edit box](#)
- [00603 Styling Master List](#)
- [00604 Trying to ease the RubyGems pain](#)
- [00605 Shoes snapshot](#)
- [00606 shoes gem alternate repo's](#)
- [00607 http post in shoes](#)

7. Assignment

- [00701 Assignment 1 twitter client \(reader\)](#)
- [00702 Assignment 2 footracer \(mini-footracer-1st.rb, mini-footracer-2nd.rb\)](#)
- [00703 Assignment 3 Mini Adventure Game GUI Part \(sample52.rb, sample52-render.rb\)](#)
- [00704 Assignment 4 Pong in Shoes \(sample58.rb\)](#)
- [00705 Assignment 5 Riddles in Shoes \(sample67.rb\)](#)
- [00706 Assignment 6 Dog Hunts Sheep Game \(sample68.rb\)](#)

8. [00800 Relevant web sites \(Links\) \(sample62.rb\)](#)

9. [00900 Appendix](#)

10. [01000 Acknowledgment](#)

11. [01100 Fancy Gallery \(gallery1.rb, gallery2.rb, gallery3.rb, gallery4.rb, gallery4-1.rb, gallery5.rb\)](#)

12. Built-in Samples

- [01201 simple-accordion \(simple-accordion.rb\)](#)
- [01202 simple-calc \(simple-calc.rb, sample61.rb\)](#)
- [01203 simple-menu \(simple-menu-r1.rb\)](#)

13. [01300 Trivia \(sample91.rb, sample92.rb, sample93.rb\)](#)

Change log:

Look at [changelog.mdown](#).

To do list:

- Add search function.
- Improve mkpdf.rb to form more beautifully.
- Improve the browser feature to resizable.
- Improve mkbightml.rb for creating PDF file.

Let's enjoy Ruby and Shoes programming!!

:-D

ashbb

Introduction

Shoes is a tiny cross-platform graphics and windowing toolkit for the Ruby programming language written by [why](#).

All sample programs and data files in this tutorial can be downloaded from [here](#).

Some sample programs are taken from chapter 8 of [NKS](#) (Nobody Knows Shoes, the first publicly available manual for Shoes).

Download Shoes

Download Shoes from the [project's web site](#) and read the [installation instructions](#) for your platform.

We use Shoes 2 (Raisins, 0.r1134) in this tutorial.

First step

A good starting point for working with Shoes is the [tutorial written by why](#).

If you want to experiment, you can copy and paste the 16 sample programs and run them one by one. Don't worry if you don't understand the code yet, just run it and see what happens to the app windows. Although the tutorial does have screenshots, running the samples and maybe playing around with them is a good start to Shoes programming. So do it now, it will help you a lot in understanding this tutorial you are reading right now.

Tip: Quick launch Shoes from the shell

Add the following line to your .bashrc (create it if the file doesn't exist):

```
alias shoes='/Applications/Shoes.app/Contents/MacOS/shoes'
# your path might be different....
```

Now you can launch a Shoes app from a terminal window like this, thus avoiding going through the open file dialog:

```
shoes sample1.rb
```

This tip was provided by George Thompson - Saturday, 15 November 2008, POIRPWSC101-11

Tip: Quick launch Shoes from TextMate

A video that shows how to launch Shoes from within TextMate(<http://samuraicoder.net/shoes.mov>) # Note: Unfortunately this link does not seem to be working right now.

This tip was provided by Takaaki Kato - Saturday, 15 November 2008, POIRPWSC101-11

Tip: Quick launch Shoes in Windows

Create a shortcut in the "Tool Bar".

shoes_launcher.jpg



This tip was provided by Victor Goff - Saturday, 15 November 2008, POIRPWSC101-11

Tip: For users of Ubuntu 8.10

The documentation is no longer correct. It was for Ubuntu 8.04. For Ubuntu 8.10, the required libraries are libvlc-dev and libvlccore-dev instead of libvlc0-dev.

This tip was provided by Jose Carlos Monteiro - Saturday, 15 November 2008, POIRPWSC101-11

Information about Kate

I use Kate (Kubuntu 8.04 text editor) and it has support for a number of programming languages and a feature that opens a terminal at the bottom of the text area (this height can be adjusted too).

Another thing I like about this is that the terminal is* attached to the editor *and if you've moved to another directory just close the terminal and open it to be at the new directory.

You may use if you like.

This info was provided by Dave Lilley - Mar 11th, 2009

Birds-eye view (Survey of basic features)

Concept

Shoes is a tiny graphics toolkit. It's simple and was designed to be easy to use! Therefore Shoes doesn't have elements like tabbed controls, toolbars or horizontal scrollbars. Instead they can be simulated with images. There are only ten essential methods to know to get going with Shoes.

No. 1: para

banner : Character size 48 pixels

title : 34 pixels

subtitle : 26 pixels

tagline : 18 pixels

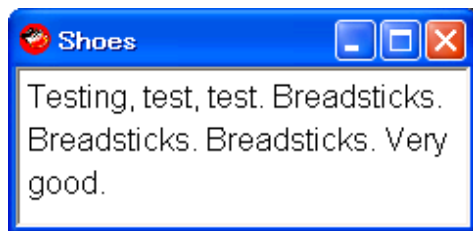
caption : 14 pixels

para (paragraph) : 12 pixels

inscription : 10 pixels

```
# sample1.rb
Shoes.app :width => 230, :height => 80 do
  para 'Testing, test, test. ',
    'Breadsticks. ',
    'Breadsticks. ',
    'Breadsticks. ',
    'Very good.'
end
```

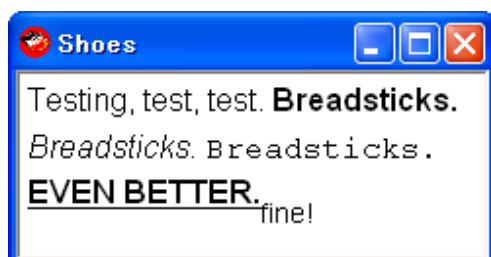
sample1.png



strong : bold
 em (emphasized) : italics
 code : monospace font
 ins (inserted) : single underline
 sub (subscript) : lowers the text by 10 pixels, x-small font

```
# sample2.rb
Shoes.app :width => 240, :height => 95 do
  para 'Testing, test, test. ',
    strong('Breadsticks. '),
    em('Breadsticks. '),
    code('Breadsticks. '),
    strong(ins('EVEN BETTER. ')),
    sub('fine!')
end
```

sample2.png



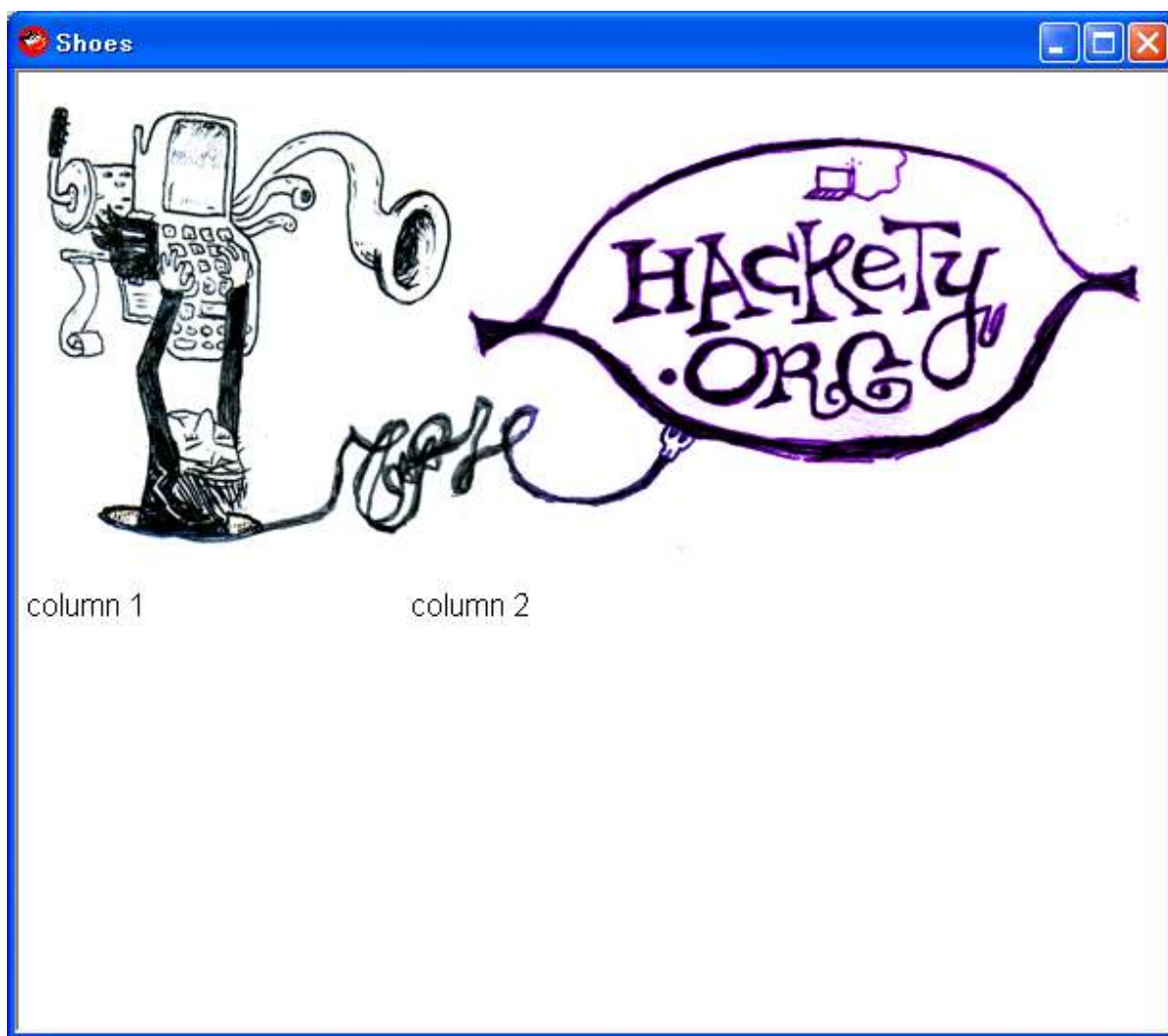
Nos. 2 & 3: stack and flow

At first, please read the following web page: <http://github.com/why/shoes/wikis/stacksandflows>

Note Instead of using the code from the above document which uses the deprecated Shoes#text method, please run the following sample code (and keep in mind to change the path to the image file):

```
# sample3.rb
Shoes.app do
  stack do
    image "http://hackety.org/images/hackety-org-header.png"
  end
  stack :width => 200 do
    para "column 1"
  end
  stack :width => -200 do
    para "column 2"
  end
end
end
```

sample3.png



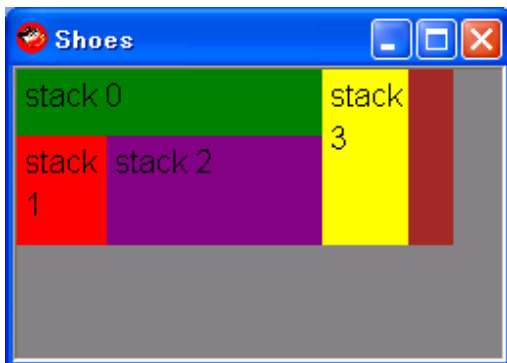
A more complex example:

```
# sample4.rb
Shoes.app :width => 250, :height => 150 do
  background gray
  flow :width => "90%" do
    background brown

    flow :width => "70%" do
      background purple
      stack do
        background green
        para "stack 0"
      end
      stack :width => "30%" do
        background red
        para "stack 1"
      end
      stack :width => "-30%" do
        background blue
        para "stack 2"
      end
    end
  end

  stack :width => "20%" do
    background yellow
    para "stack 3"
  end
end
end
```

sample4.png



No. 4: button

The method call

```
button("Press Me")
```

creates a new button whereas

```
button("Press Me"){ alert("clicked")}
```

runs the associated block when the button is clicked. Finally

```
button("Press Me", :left => 50, :top => 20)
```

will place the button at the coordinates (50, 20).
That's all there is to know about buttons.

```
# sample5.rb
Shoes.app :width => 200, :height => 50 do
  button("Press Me", :left => 50, :top => 20) do
    alert("clicked")
  end
end
```

sample5.png

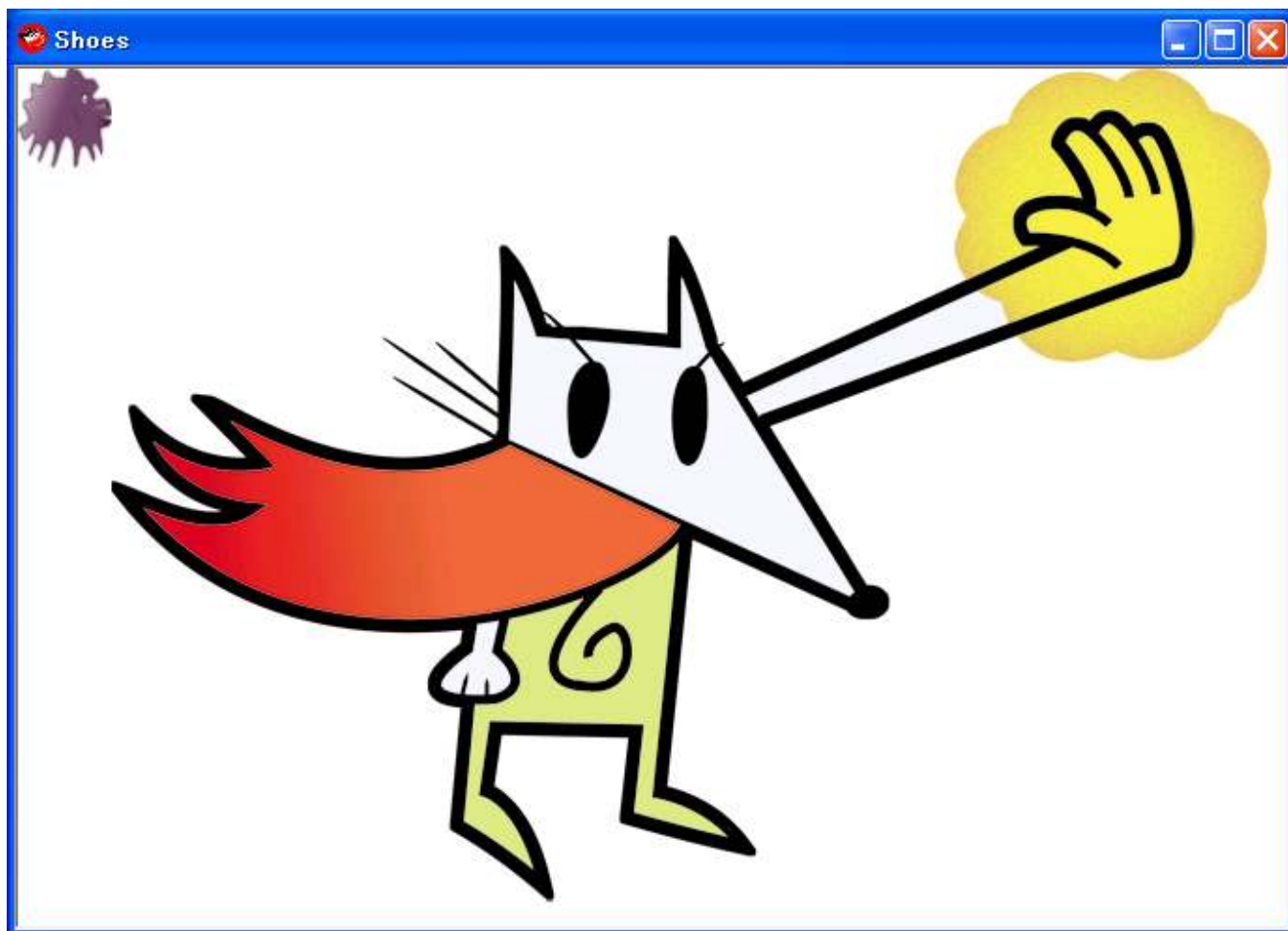


No. 5: image

An image is a picture in PNG, JPEG or GIF format. We can load it from a local directory path or an URL.

```
# sample6.rb
Shoes.app :width => 680, :height => 460 do
  image '../images/loogink.png'
  image "http://hacktyhack.net/images/design/Hacky-Mouse-Hand.png"
end
```

sample6.png



No. 6: edit_line

Edit boxes are wide, rectangular boxes for entering text (think of the HTML "textarea" tag). Edit lines are slender, little boxes for entering one line of text (think of the HTML "input" tag).

```
# sample7.rb
Shoes.app :width => 250, :height => 300 do
  stack do
    @msg = para 'Hello'
    @el = edit_line "We love Ruby."
    button('ok'){ @msg.text = @el.text}
    @eb = edit_box "We love Shoes."
    button('ok'){ @msg.text = @eb.text}
  end
end
```

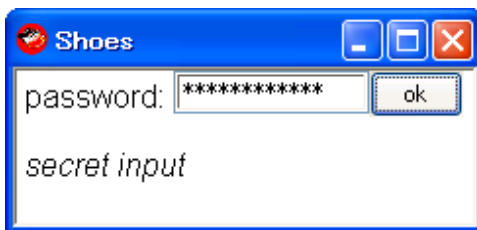
sample7.png



We can use the attribute `:secret` for creating password fields with `edit_line` (think of `type="password"` in HTML `"input"` tags):

```
# sample7-1.rb
Shoes.app :width => 235, :height => 80 do
  para 'password: '
  @el = edit_line :width => 100, :secret => true
  button('ok'){@input.replace em(@el.text)}
  @input = para ''
end
```

sample7-1.png



No. 7: link

We have three different ways to include hyperlinks in our Shoes app:

```
# sample8.rb
Shoes.app :width => 250, :height => 60 do
  para link('RubyLearning.org'){visit "http://www.rubylearning.org/"}
  para link('Google', :click => "http://google.com")
  image '../images/loogink.png', :click => "http://shoooes.net/"
end
```

sample8.png



No. 8: background

Backgrounds and borders are both just patterns. They are actual elements, not styles. A pattern is made with a color, a gradient or an image.

```
# sample9.rb
Shoes.app :width => 200, :height => 140 do
  background '#FF9900'
  background rgb(192, 128, 0), :left => 40
  background gray(0.6), :left => 80
  background red, :left => 120
  background '#FAD'..'#ADD', :left => 160
  border '../images/loogink.png', :strokewidth => 15
end
```

sample9.png



In NKS (Nobody Knows Shoes), you just give the background a radius like this

```
Background blue, :radius => 12 <br>
```

This is now obsolete, so please use `:curve` instead of `:radius`. We can also define an `:angle` for the gradient.

```
# sample10.rb
Shoes.app :width => 200, :height => 70 do
  background "#D0A"..'darkorange.to_s', :angle => 45, :curve => 30
end
```

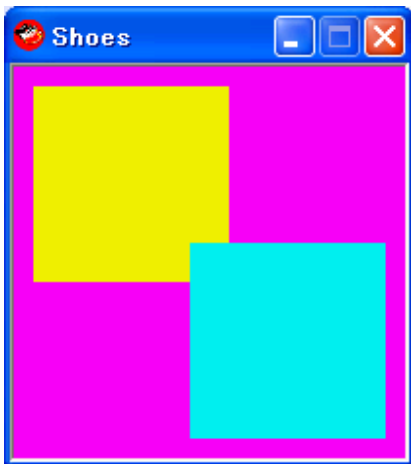
sample10.png



Changing the background color is really easy in Shoes:

```
#sample57.rb
Shoes.app :width => 200, :height => 200, :resizable => false do
  @s1 = flow :left => 10, :top => 10, :width => 100, :height => 100
  @s2 = flow :left => 90, :top => 90, :width => 100, :height => 100
  animate 24 do|i|
    @s1.background rgb(i, i, 0)
    @s2.background rgb(0, i, i)
    background rgb(i, 0, i)
  end
end
```

sample57.png



[changing color of background](#)

No. 9: Shoes.url

A Shoes "app" object is a single window running code from a Shoes URL. When you switch Shoes URLs, a new "app" object is created. From the user viewpoint, it behaves just like a web page.

```

# sample11.rb
class PhotoFrame < Shoes
  url '/', :index
  url '/loogink', :loogink
  url '/cy', :cy

  def index
    eval(['loogink', 'cy'][rand 2])
  end

  def loogink
    background tomato
    image '../images/loogink.png', :left => 70, :top => 10
    para "\n" * 3
    para strong 'She is Loogink. :)', :stroke => white
    para '->', link(strong('Cy'), :click => '/cy')
  end

  def cy
    background paleturquoise
    image '../images/cy.png', :left => 70, :top => 10
    para "\n" * 3
    para strong 'He is Cy. :)', :stroke => white
    para ' ->', link(strong('loogink'), :click => '/loogink')
  end
end

Shoes.app :width => 200, :height => 120, :title => 'Photo Frame'

```

sample11.png



No. 10: clear

The method "clear" wipes the slot. It also takes an optional block that will be used to replace the contents of the slot.

```
# sample12.rb
Shoes.app :title => 'RC', :width => 100, :height => 80 do
  def random_creatures
    background rgb rand(256), rand(256), rand(256)
    name = %w[loogink cy yar kamome shaha][rand 5]
    image '../images/' + name + '.png', :left => 30, :top => 10
  end

  random_creatures

  every(5){clear{random_creatures}}
end
```

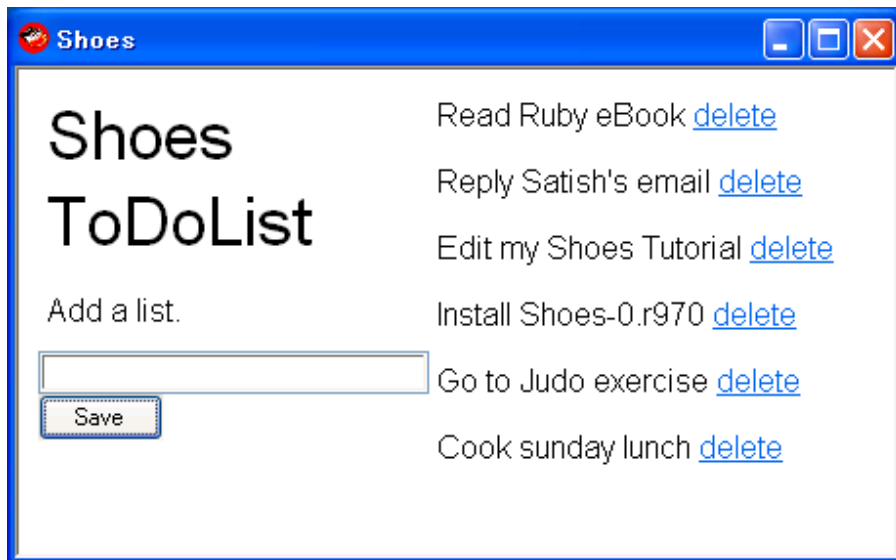
sample12.png



The method "append" and "remove" are also very useful for manipulating the contents of a slot.

```
# sample13.rb
Shoes.app :width => 450, :height => 250 do
  stack :margin => 10, :width => 200 do
    subtitle 'Shoes ToDoList'
    para 'Add a list.'
    @add = edit_line
    button 'Save' do
      @notes.append do
        para @add.text, ' ', link('delete'){|e| e.parent.remove}
      end
      @add.text = ''
    end
  end
  @notes = stack :margin => 10, :width => -200
end
```

sample13.png



Tips for creating your own Shoes apps

Open the built-in manual and the console window

To open the built-in manual, type the following in your console or terminal window:

```
shoes -m
or
shoes --manual
```

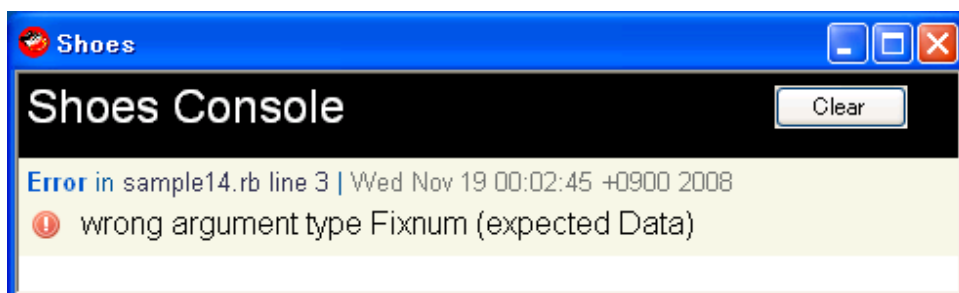
The keyboard shortcut `Alt + ?` ('⌘M' on a Mac) opens the manual from inside any Shoes app window.

Alternatively you can also access the manual from the menu like this:

<http://shoooes.net/manuals/>

To open the Shoes console window, type `Alt + /` ('⌘/' on a Mac) in any Shoes application window.

shoes_console.png

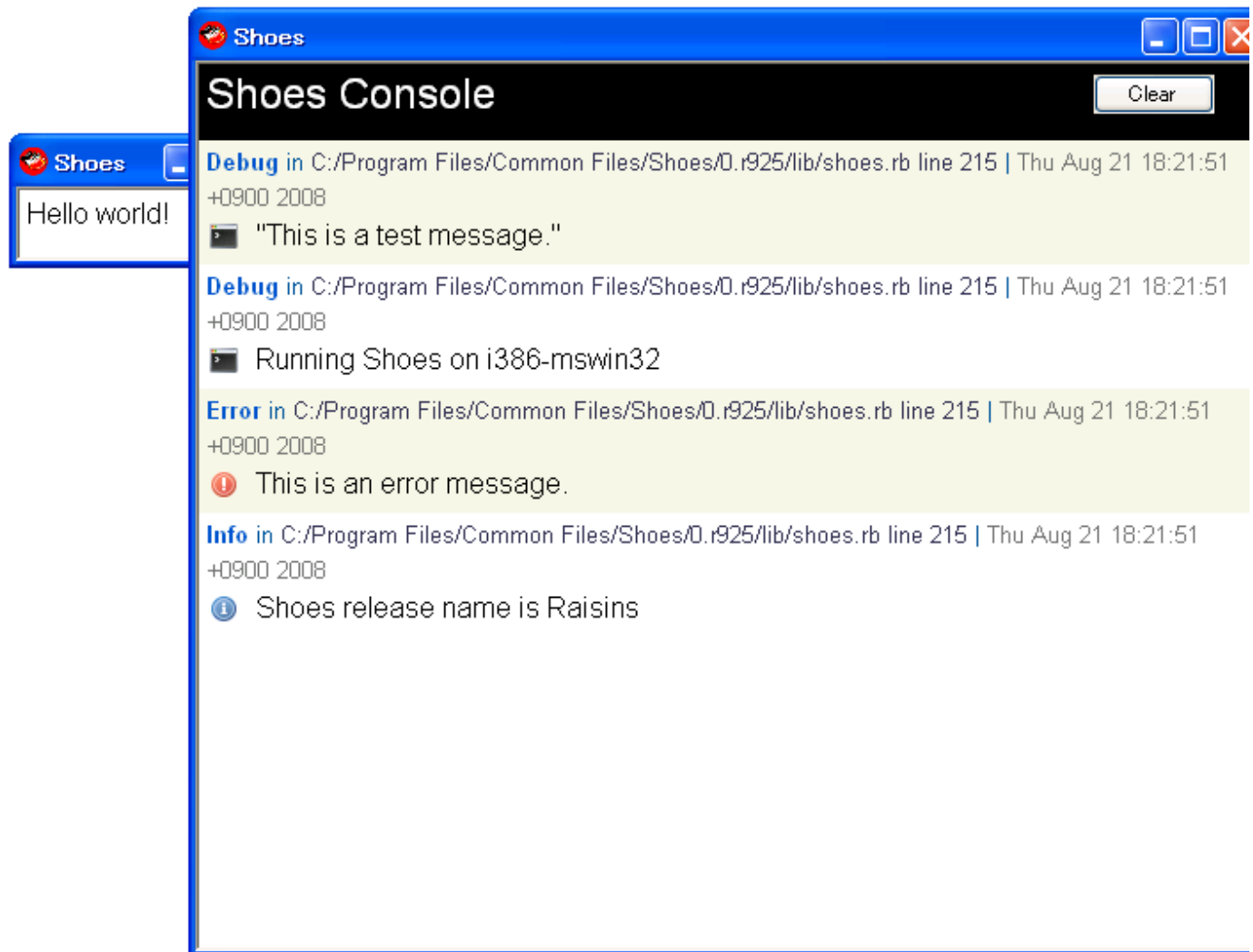


Output messages on the Shoes console window

To debug your application, you can send messages to the Shoes console window.

```
# sample15.rb
Shoes.app :width => 150, :height => 40 do
  para 'Hello world!'
  Shoes.p 'This is a test message.'
  debug 'Running Shoes on ' + RUBY_PLATFORM
  error 'This is an error message.'
  info 'Shoes release name is ' + Shoes::RELEASE_NAME
end
```

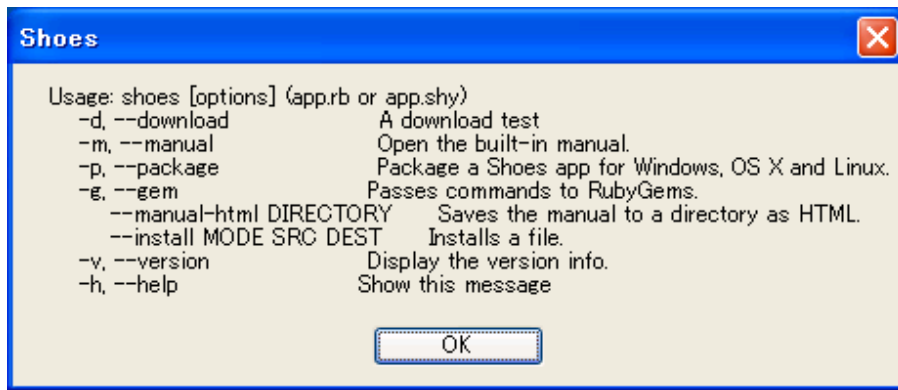
sample15.png



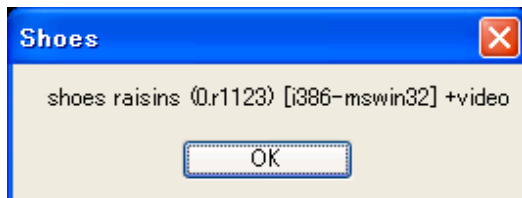
shoes --help

Type the following in your console or terminal window to access Shoes' help system:

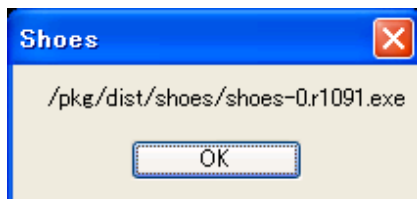
```
shoes -h
or
shoes --help
shoes_help.png
```

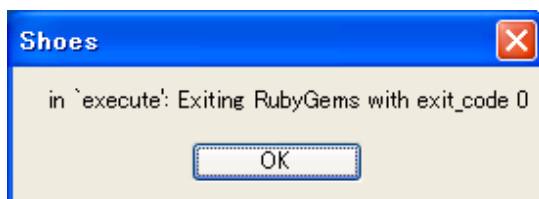
shoes_version.png



shoes_download_test.png

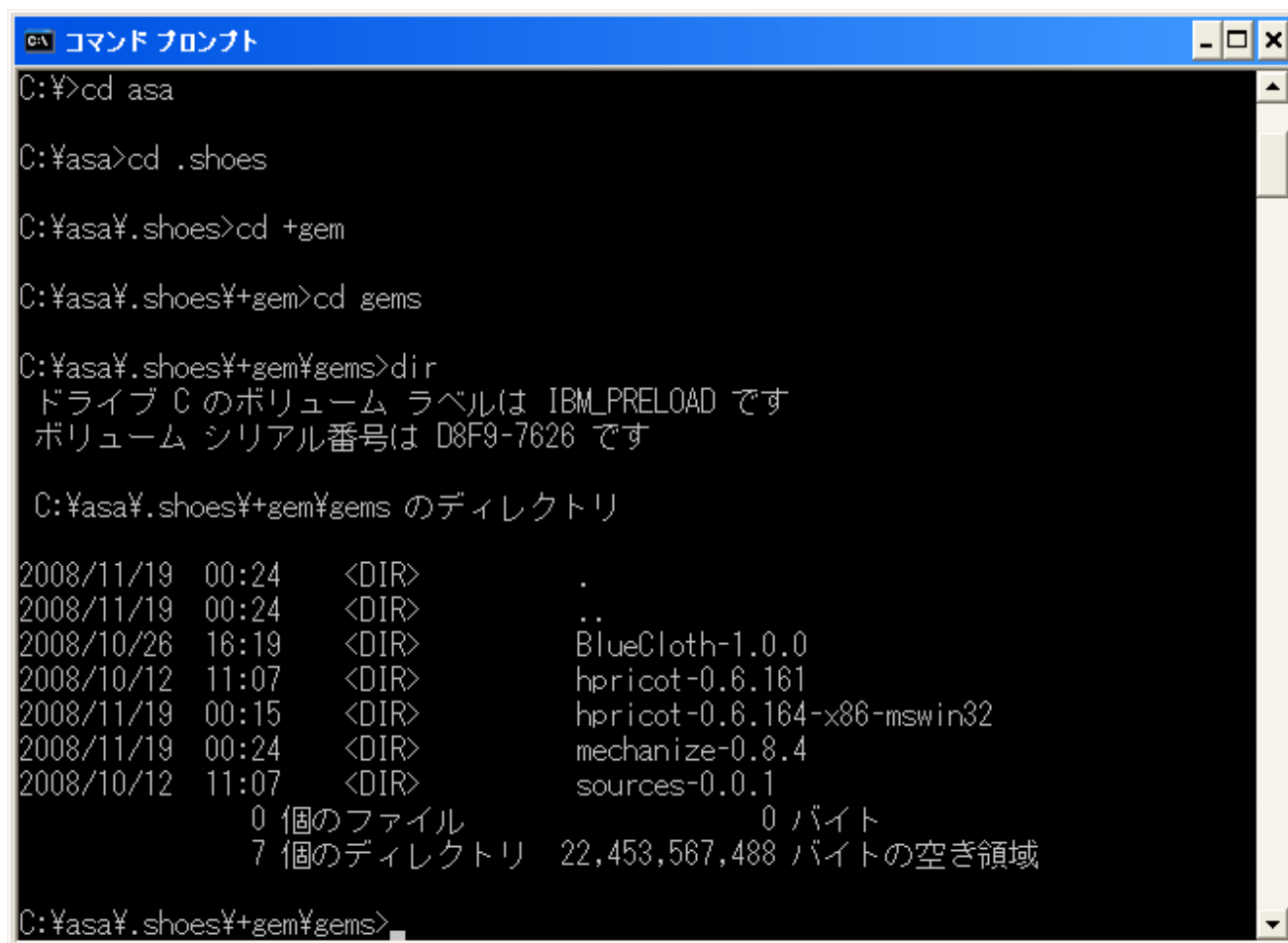


shoes_gem.png



shoes -g install hpricot
 shoes -g install mechanize
 This will install gems for Shoes only, your regular Ruby installation is not affected by this!

shoes_gem-1.png



```

C:\ コマンド プロンプト
C:¥>cd asa
C:¥asa>cd .shoes
C:¥asa¥.shoes>cd +gem
C:¥asa¥.shoes¥+gem>cd gems
C:¥asa¥.shoes¥+gem¥gems>dir
ドライブ C のボリューム ラベルは IBM_LPRELOAD です
ボリューム シリアル番号は D8F9-7626 です

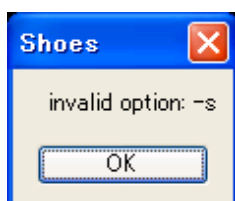
C:¥asa¥.shoes¥+gem¥gems のディレクトリ

2008/11/19  00:24    <DIR>          .
2008/11/19  00:24    <DIR>          ..
2008/10/26  16:19    <DIR>          BlueCloth-1.0.0
2008/10/12  11:07    <DIR>          hpricot-0.6.161
2008/11/19  00:15    <DIR>          hpricot-0.6.164-x86-mswin32
2008/11/19  00:24    <DIR>          mechanize-0.8.4
2008/10/12  11:07    <DIR>          sources-0.0.1
               0 個のファイル                0 バイト
               7 個のディレクトリ  22,453,567,488 バイトの空き領域

C:¥asa¥.shoes¥+gem¥gems>

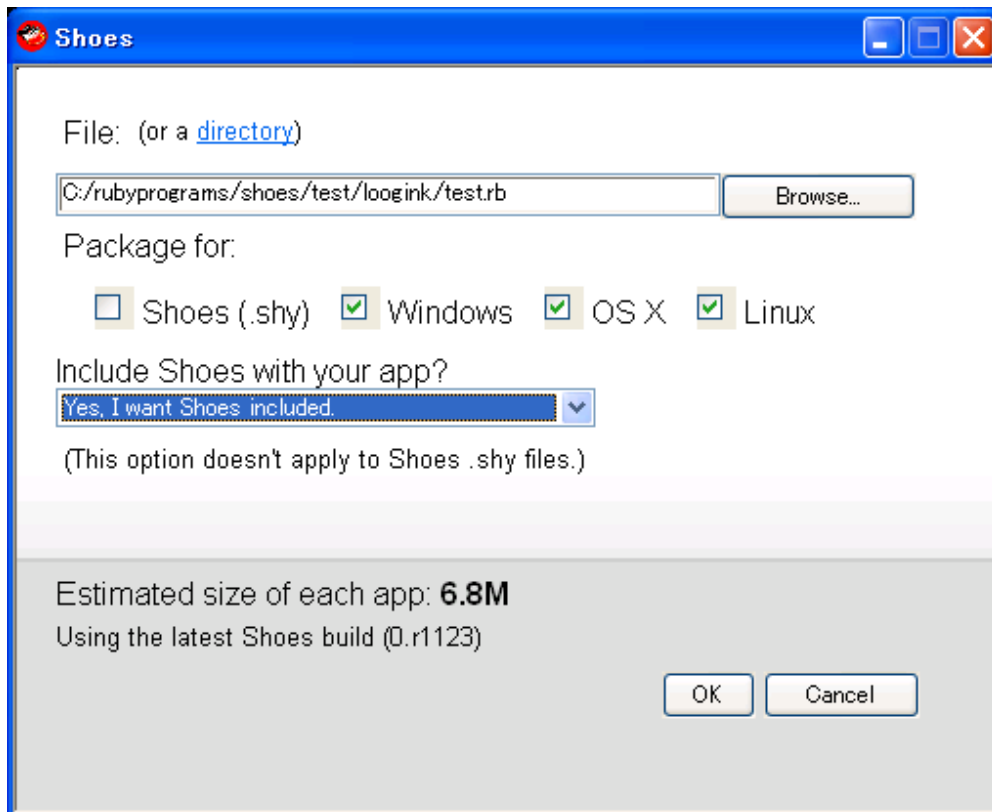
```

shoes_shy-1.png



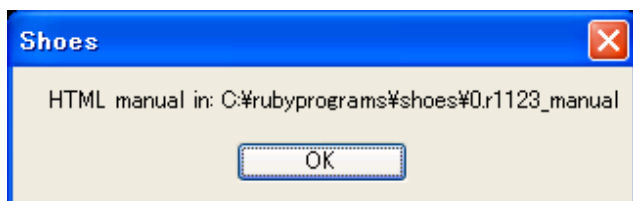
Shoes version newer than 0.r1057 wil display the above warning since this functionality is now part of 'shoes --package'.

shoes_package.png



Here you can select a file, which will then get packaged for Windows, Linux or Mac OS X. Alternatively you can also package it as a .shy file for Shoes.

shoes_manual-html.png



To create an HTML version of the Shoes manual you can do the following:

```
C:\>cd C:\Program Files\Common Files\Shoes\0.r1123
C:\Program Files\Common Files\Shoes\0.r1123>shoes --manual-html C:\rubyprograms\shoes\0.r1123_manual
```

shoes_manual-html2.png

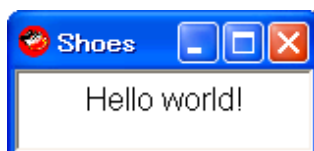


App object and coding style

A Shoes App object is a single window running code at a Shoes URL. When you switch Shoes URLs, a new App object is created. The application itself is a flow. There are four Shoes App object coding styles:

```
# sample16.rb
Shoes.app :width => 150, :height => 40 do
  para 'Hello world!', :align => 'center'
end
```

sample16.png



```
# sample17.rb
class Hello < Shoes
  url '/', :index

  def index
    para 'Hello world!', :align => 'center'
  end
end

Shoes.app :width => 150, :height => 40
```

sample17.png This is essentially the same as sample16 above.

```
# sample18.rb
class Shoes::Hello < Shoes::Widget
  def initialize
    para 'Hello world!', :align => 'center'
  end
end

Shoes.app :width => 150, :height => 40 do
  hello
end
```

sample18.png Another way of recreating sample16.

```
# sample47.rb
blk = proc{para 'Hello world!', :align => 'center'}

Shoes.app :width => 150, :height => 40, &blk
```

sample47.png This also will give you the same result as sample16.

More information

Open question: When to use one coding style over another?

This question was asked by Jose Carlos Monteiro - Saturday, 15 November 2008, POIRPWSC101-11

As far as we know, there is no "official" guideline, so here's a subjective approach:

a) Sample 16 style

This is the easiest and most simple to read and therefore ideal for beginners.

b) Sample 17 style

This is a special style for using Shoes URLs.

Please refer to [No. 9: Shoes.url](#)

c) Sample 18 style

This is a special style for using Shoes' Widget class.

Please refer to [the Widget class](#)

A more thorough explanation can be found [here](#)

d) Sample 47 style

This style is used when you want to work with a block or proc object.
Please refer to [Open a new app window \(Another example\)](#)

NOTE

Please don't worry if you don't understand all of the above right now, it's best to learn Shoes step by step. Also keep in mind that Shoes is still quite young and therefore actively developed, so certain behaviors are bound to change.

Built-in Constants and methods

Built-in Constants:

Shoes::RELEASE_NAME

Shoes::RELEASE_ID

Shoes::REVISION

Shoes::FONTS

Built-in methods:

These methods can be used anywhere throughout a Shoes program:

alert, ask, ask_color, ask_open_file, ask_save_file, ask_open_folder, ask_save_folder, confirm, debug, error, exit, font, gradient, gray, info, rgb, warn

Read the built-in manual: -> Shoes -> Built-in section.

Scope: A tip about using YAML files

```
# sample19.rb
require 'yaml'

Shoes.app :width => 200, :height => 100 do
  Gang = Struct.new :name, :country
  gangs = YAML.load_file(Dir.pwd + '/gangs.yml')
  gangs.each{|g| para g.name, g.country, "\n"}
end
```

sample19.png



The top-level namespace in any Shoes app is Shoes
so in the sample

```
Gang = Struct.new :name, :country <br>
```

we actually create a Shoes::Gang struct, not a Gang struct.
To achieve the desired behavior, we will have to modify the statement like this (see sample19-1.rb).

```
::Gang = Struct.new :name, :country <br>
```

Here you can see the updated example:

```
# sample19-1.rb
require 'yaml'

Shoes.app :width => 200, :height => 100 do
  ::Gang = Struct.new :name, :country
  gangs = YAML.load_file(Dir.pwd + '/gangs.yml')
  gangs.each{|g| para g.name, g.country, "\n"}
end
```

sample19-1.png



keypress, mouse and clipboard

A Shoes program can react to mouse events and interact with the system's clipboard.

```
# sample20.rb
Shoes.app :title => 'Sorter', :width => 180, :height => 80 do
  background gradient powderblue, royalblue
  msg = para '', :size => 8

  yar = image('../images/yar.png', :left => 60, :top => 18).click do
    self.clipboard = self.clipboard.sort unless self.clipboard.nil?
    yar.transform :center
    a = animate(24) do |i|
      yar.rotate -15
      a.stop if i > 22
    end
  end
  yar.hover{msg.text = strong('Click Yar. She sorts clipboard text!')}
  yar.leave{msg.text = ''}
end
```

sample20.png



Let's see how this program manipulates the clipboard

Before:

Creatures name list is:

looginkff
cy
kamome
yar
shaha

Copy the above list into the system clipboard.

Click Yar and she will rotate (*1).

Paste the clipboard text into the place you want.

*1: With Shoes-0.r925, Yar rotates as expected, but with Shoes-0.r970, Yar rotates when the mouse moves out of the Shoes window. This behavior is a bug. It has been fixed in Shoes-0.r1057.

After:

Creatures name list is:

cy
kamome
loogink
shaha
yar

How to work with keypress events:

```
# sample21.rb
Shoes.app :width => 250, :height => 40 do
  @info = para 'NO KEY is PRESSED.'
  keypress{|key| @info.text = "#{key.inspect} was PRESSED."}
end
```

sample21.png



ATTENTION

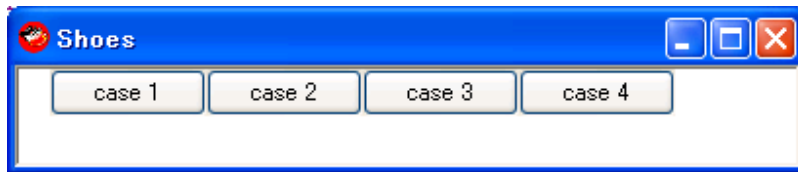
Please be aware of the following behavior which may be caused by a bug in Shoes:

In the following code snippet, cases 1, 3 and 4 will crash. Case 2 works because it creates a new empty string object ("") which it assigns to `self.clipboard` instead of `p2.text` in case the latter is empty.

This problem was discovered in the code of [rotten.rb](#) created by [Michael Kohl](#). It does not seem to affect Shoes on Mac OS X where the program is developed.


```
# sample65.rb
Shoes.app :width => 400, :height => 50 do
  p1 = para
  p2 = para
  p3 = para
  p4 = para
  button('case 1'){self.clipboard = p1.text}
  button('case 2'){self.clipboard = p2.text.eql?('') ? '' : p2.text }
  button('case 3'){self.clipboard = p3.text.eql?('') ? p3.text : p3.text }
  button('case 4'){self.clipboard = p4.text.eql?('') ? '' : p4.text }
end
```

sample65.png



The Widget class

A custom Shoes widget is set up by inheriting from the Widget class. Shoes then creates a method using the lowercased name of the class which is used in your app.

```
# sample22.rb
class Shoes::Answer < Shoes::Widget
  attr_reader :mark
  def initialize word
    para word
    @mark = image('../images/loogink.png', :width => 20, :height => 20).
  end
end

Shoes.app :width => 200, :height => 130 do
  stack :width => 0.5 do
    background palegreen
    para '1. apple'
    ans = answer '2. tomato'
    para '3. orange'
    button('Ans.'){ans.mark.toggle}
  end
  stack :width => 0.5 do
    background lightsteelblue
    para '1. cat'
    para '2. dog'
    ans = answer '3. bird'
    button('Ans.'){ans.mark.toggle}
  end
end
```

sample22.png



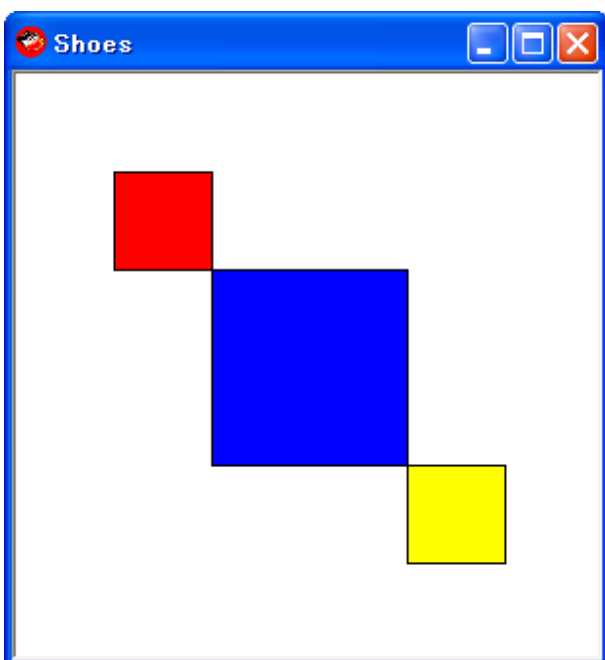
One more example

We can use `:left` and `:top` without definition in the custom class attributes, because they are inherited from `Shoes::Widget`.

```
# sample49.rb
class Shoes::widgy < Shoes::widget
  def initialize opts = {}
    size = opts[:size] || 50
    fill opts[:color] || yellow
    rect 0, 0, size, size
  end
end

Shoes.app :width => 300, :height => 300 do
  w1 = widgy :left => 50, :top => 50, :color => red
  w2 = widgy :left => 50, :top => 50, :color => blue, :size => 100
  w2.move 100, 100
  w3 = widgy
  w3.left = 200
  w3.top = 200
end
```

sample49.png



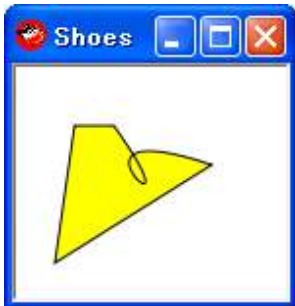
Original idea was provided by Emanuele Carnevale, in the Shoes ML.

shape

We can create arbitrary shapes, starting at the coordinates (left, top).

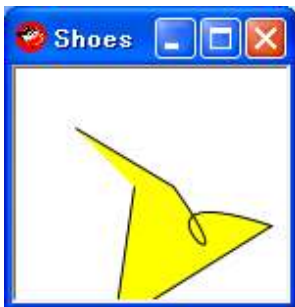
```
# sample23.rb
Shoes.app :width => 140, :height => 120 do
  fill yellow
  shape :left => 30, :top => 30 do
    line_to 50, 30
    curve_to 100, 100, 10, 20, 100, 50
    line_to 20, 100
    line_to 30, 30
  end
end
```

sample23.png



There seems to be a bug with Shoes-0.r925. Please note the difference in the two screenshots, where the above was taken using Shoes-0.r905 and the second one using Shoes-0.r925.

sample23-1.png



From Shoes version 0.r-970 it seems to be working again, although the result doesn't exactly look like the first image. After Shoes-0.r970 we are back to the 0.r-925 behavior.

mask

The use of a masking layer is described by Shoes' author `_why` in the following article:

[Cut Holes In Shoes And Get A Mask](#)

```
# sample24.rb
Shoes.app :width => 160, :height => 80 do
  def mask_words
    strokewidth 4
    160.times do |i|
      stroke send COLORS.keys[rand COLORS.keys.length]
      line i * 4 - 50, 0, i * 4, 80
    end
    mask :margin => 4 do
      title strong 'Shoes'
    end
  end
end

mask_words
every 3 do
  clear{ mask_words }
end
end
```

sample24.png



Drawing directly onto images

It's possible to add elements to an image.

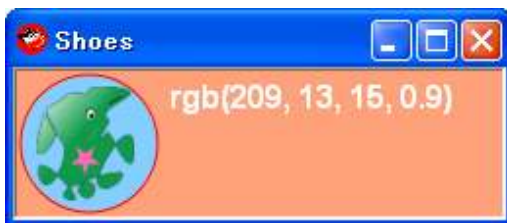
In the following example Cy (the green creature) had a star added to it.

```
# sample25.rb
Shoes.app :width => 250, :height => 76 do
  background lightsalmon
  icon = image :width => 74, :height => 74 do
    oval :width => 70, :height => 70, :fill => lightskyblue,
        :stroke => red, :left => 2, :top => 2

  end
  icon.image '../images/cy.png', :left => 10, :top => 8
  icon.star 35, 45, 5, 8, 3, :fill => hotpink, :stroke => nil
  msg = para '', :stroke => white

  icon.hover do
    @a = animate do
      button, left, top = self.mouse
      msg.replace strong icon[left, top]
    end
  end
  icon.leave do
    @a.stop
    msg.replace ''
  end
end
```

sample25.png



Please note that if you are using Shoes-0.r970, you need to move the mouse off the image once. After that it will work as expected.

With Shoes-0.r1057, there is no need to do this.

Style

We can change the style of an element with the style method, whereas calling the method without arguments returns a hash of the styles presently applied to the element.

More information can be found in section 6.3, Styling Master List

```

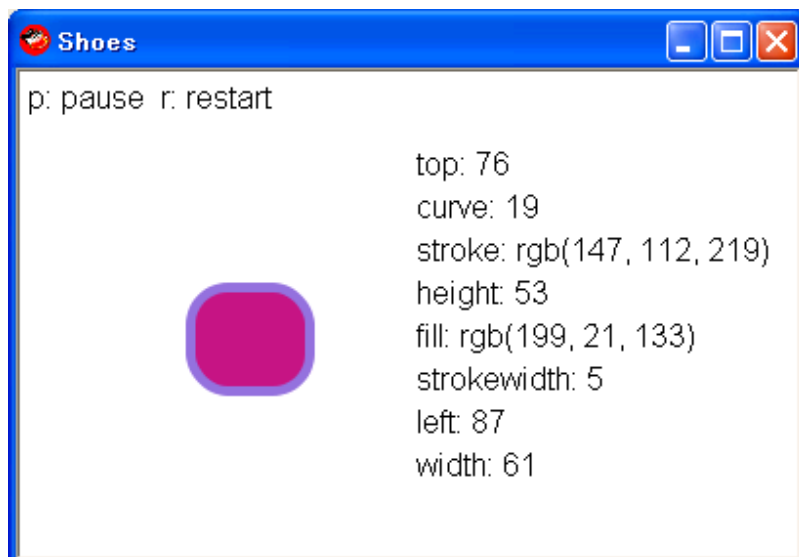
# sample26.rb
Shoes.app :width => 400, :height => 250 do
  def sampling
    stack(:width => 1.0){para 'p: pause r: restart'}
    #stack(:width => 0.5){@o = oval 0, 0, 50}
    stack(:width => 0.5){@r = rect 0, 0, 50, 50, 10}
    stack(:width => 0.5){@p = para ''}

    @a = every(1) do
      @r.style :width => 10 + rand(100), :height => 10 + rand(100),
              :curve => rand(20),
              :fill => send( COLORS.keys[rand COLORS.keys.length] ),
              :strokewidth => rand(10),
              :stroke => send( COLORS.keys[rand COLORS.keys.length])
      @r.move rand(100), rand(100)
      @p.replace @r.style.to_a.map{|e| e.join(': ')} .join("\n")
    end
  end

  sampling
  keypress do |k|
    case k
    when 'p'
      @a.stop
    when 'r'
      @a.stop if @a
      clear{sampling}
    else
    end
  end
end
end

```

sample26.png



We can style an entire class of elements at a time like this:

```
# sample56.rb
Shoes.app :width => 200, :height => 160 do
  style Para, :align => 'center', :stroke => pink, :size => 30
  stack do
    para "hello"
    para "hello", :size => 10, :stroke => red
    para "hello"
  end
end
```

sample56.png



Shoes.setup

If your Shoes app requires some libraries, this might be useful. See the following information:

[Clearing Up The Whole Shoes And RubyGems Deal](#)

```
# sample27.rb
Shoes.setup do
  gem 'something'
end

Shoes.app do
  para require 'something'
end
```

sample27.png



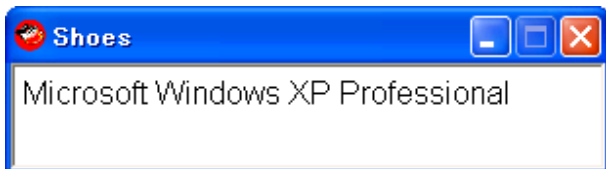
One more example

```
# sample50.rb
Shoes.setup do
  gem 'sys-uname'
end

require 'sys/uname'
include Sys

Shoes.app :width => 300, :height => 50, :resizable => false do
  @platform = para Uname.sysname
end
```

sample50.png



Original snippet was written by Massimiliano in a personal mail discussion.

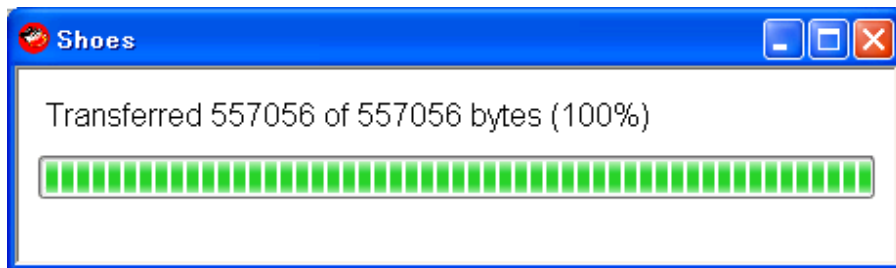
Downloader

This method starts a download thread running in the background and displays a handy progress bar. Please keep in mind that the 'percent' and 'length' methods don't seem to work well at this time, whereas 'transferred' and 'fraction' do.

```
# sample28.rb
Shoes.app :width => 450, :height => 100 do
  stack :margin => 10 do
    url = 'http://shoooes.net/dist/shoes-0.r1057.exe'
    status = para "Downloading #{url}"
    p = progress :width => 1.0

    download url,
      :save => Dir.pwd + '/' + File.basename(url),
      :start => proc{|dl| status.text = 'Connecting...'},
      :progress => proc{|dl|
        status.text = "Transferred #{dl.transferred} of #{dl.length} byt
        p.fraction = dl.percent * 0.01},
      :finish => proc{|dl| status.text = 'Download finished'},
      :error => proc{|dl, err| status.text = "Error: #{err}" }
  end
end
```

sample28.png



Shoes includes the Hpricot library for parsing HTML.
For more information about downloader, please consult the Shoes manual.

Assign Shoes URLs dynamically

We can use regular expressions to assign Shoes URLs dynamically.
Shoes passes the match data to the method as the argument.
The following sample code modifies sample11.rb from before.

```
# sample29.rb
class PhotoFrame < Shoes
  url '/', :index
  url '/(.+)', :index

  Creature = Struct.new :name, :sex, :wallpaper
  @@c = []
  @@c << Creature.new('loogink', 'She', 'tomato')
  @@c << Creature.new('cy', 'He', 'paleturquoise')

  def index n = rand(2)
    n = n.to_i
    background eval(@@c[n].wallpaper)
    image '../images/' + @@c[n].name + '.png', :left => 70, :top => 10
    para "\n" * 3
    para strong "#{@@c[n].sex} is #{@@c[n].name.capitalize. :})", :stroke
    n = n.zero? ? 1 : 0
    para '->', link(strong(@@c[n].name.capitalize), :click => "/#{n}")
  end
end

Shoes.app :width => 200, :height => 120, :title => 'Photo Frame'
```

sample29.png
The sample29.png is almost the same as the above sample11.png.

Classes List and Colors List

We can see the colors list in the built-in manual.
We can also see them with the following sample code.

```
# sample30.rb
Shoes.app :width => 642, :height => 700, :resizable => false do
  COLORS.keys.map{|sym|sym.to_s}.sort.each do |color|
    flow :width => 160, :height => 20 do
      c = send(color)
      fill c
      rect 0, 0, 160, 20
      inscription color, :stroke => c.dark? ? white : black
    end
  end
end
```

sample30.png

aliceblue	antiquewhite	aqua	aquamarine
azure	beige	bisque	black
blanchedalmond	blue	blueviolet	brown
burlywood	cadetblue	chartreuse	chocolate
coral	cornflowerblue	cornsilk	crimson
cyan	darkblue	darkcyan	darkgoldenrod
darkgray	darkgreen	darkkhaki	darkmagenta
darkolivegreen	darkorange	darkorchid	darkred
darksalmon	darkseagreen	darkslateblue	darkslategray
darkturquoise	darkviolet	deeppink	deepskyblue
dimgray	dodgerblue	firebrick	floralwhite
forestgreen	fuchsia	gainsboro	ghostwhite
gold	goldenrod	gray	green
greenyellow	honeydew	hotpink	indianred
indigo	ivory	khaki	lavender
lavenderblush	lawngreen	lemonchiffon	lightblue
lightcoral	lightcyan	lightgoldenrodyellow	lightgreen
lightgrey	lightpink	lightsalmon	lightseagreen
lightskyblue	lightslategray	lightsteelblue	lightyellow
lime	limegreen	linen	magenta
maroon	mediumaquamarine	mediumblue	mediumorchid
mediumpurple	mediumseagreen	mediumslateblue	mediumspringgreen
mediumturquoise	mediumvioletred	midnightblue	mintcream
mistyrose	moccasin	navajowhite	navy
oldlace	olive	olivedrab	orange
orangered	orchid	palegoldenrod	palegreen
paleturquoise	palevioletred	papayawhip	peachpuff
peru	pink	plum	powderblue
purple	red	rosybrown	royalblue
saddlebrown	salmon	sandybrown	seagreen
seashell	sienna	silver	skyblue
slateblue	slategray	snow	springgreen
steelblue	tan	teal	thistle
tomato	turquoise	violet	wheat
white	whitesmoke	yellow	yellowgreen

_why is thinking about some more method related colors.

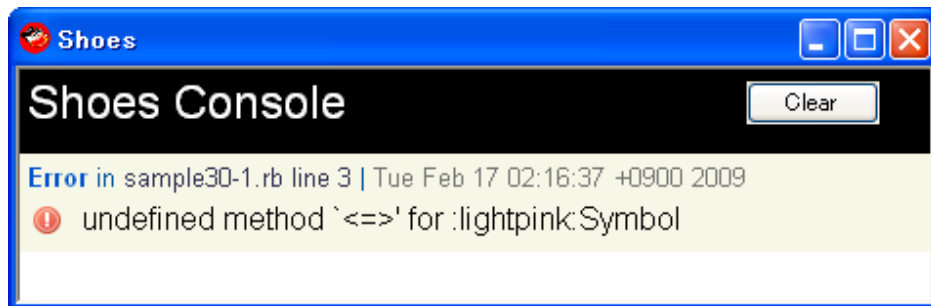
e.g. invert, dark?, light?, black?, white?, opaque?, transparent?
We might be able to get them in the near future.

Note

I used send() method in the above sample30. But I noticed an alternative way. See the below sample30-1.

```
# sample30-1.rb
Shoes.app :width => 642, :height => 700, :resizable => false do
  COLORS.sort_by{|sym,| sym.to_s}.each do |color, v|
    flow :width => 160, :height => 20 do
      fill v
      rect 0, 0, 160, 20
      inscription color, :stroke => v.dark? ? white : black
    end
  end
end
```

sample30-1.png



NOTE: There is no need to use `.to_s` in the `map` or `sort_by` method's block by right. But with Shoes 2, the below error will occur if it's ommited. I guess this is a bug of Shoes...

OMG! It's my fault. See the following.

```
C:\>ruby -v
ruby 1.8.6 (2007-09-24 patchlevel 111) [i386-mswin32]

C:\>irb --simple-prompt
>> a = [:z, :y, :m, :s]
=> [:z, :y, :m, :s]
>> a.sort
NoMethodError: undefined method `<=>' for :z:Symbol
    from (irb):2:in `sort'
    from (irb):2
>>
```

There was need to define `<=>` for Symbol class. Revised code is sample 30-2.

```
# sample30-2.rb
class ::Symbol
  def <=> other
    self.to_s <=> other.to_s
  end
end

Shoes.app :width => 642, :height => 700, :resizable => false do
  COLORS.sort_by{|sym,| sym}.each do |color, v|
    flow :width => 160, :height => 20 do
      fill v
      rect 0, 0, 160, 20
      inscription color, :stroke => v.dark? ? white : black
    end
  end
end
```

start, stop and restart

We can start something with initial conditions, then stop and restart the same thing with other conditions.

```
#sample31.rb
Shoes.app :width => 150, :height => 70 do
  def number_on_disk
    fill eval(@color)
    nostroke
    oval 0, 0, 30
    @l = para ''
    animate(3){@l.replace strong @i+=1, :stroke => white}
  end

  @color = 'blue'
  @i = 0
  @slot = flow{number_on_disk}

  button('change') do
    @slot.clear
    @color = %w(green red blue yellow)[rand(4)]
    @i = 0
    @slot.append{number_on_disk}
  end
end
```

sample31.png



Combination of image objects show/hide and mouse hover/leave

We've already learned many useful methods like show/hide and hover/leave. This tiny sample shows us a wonderful combination.

```
# sample32.rb
Shoes.app :width => 350, :height => 250, :title => 'Menus' do
  def menu items
    flow do
      items.each_with_index do |e, i|
        nostroke
        nofill
        b = image(:width => 100, :height => 21){rect(0, 0, 100, 21)}
        f = image(:width => 100, :height => 21){rect(0, 0, 100, 21, :fill)}
        b.move 0, i*23
        f.move 0, i*23
        para i, '. ', e, "\n"
        b.hover{f.show; @msg.text = strong(e)}
        b.leave{f.hide; @msg.text = ''}
      end
    end
  end

  para 'Selected: '
  @msg = para '', :stroke => green

  flow :left => 50, :top => 50 do
    para strong "what?\n"
    menu %w(apple tomato orange)
  end

  flow :left => 200, :top => 50 do
    para strong "who?\n"
    menu %w(Satoshi Krzysztof Victor Leticia Mareike)
  end
end
```

sample32.png



Here is another sample. It shows menus using many buttons.

```
# sample33.rb
Shoes.app :title => 'Button MENU', :height => 250 do
  def menu title, items, n
    button title, :align => 'center', :width => 100 do
      if @toggle[n]
        items.each{|e| @f[n].append(button(e, :align => 'center', :width
      else
        @f[n].clear
        @msg.text = ''
      end
      @toggle[n] = !@toggle[n]
    end
  end

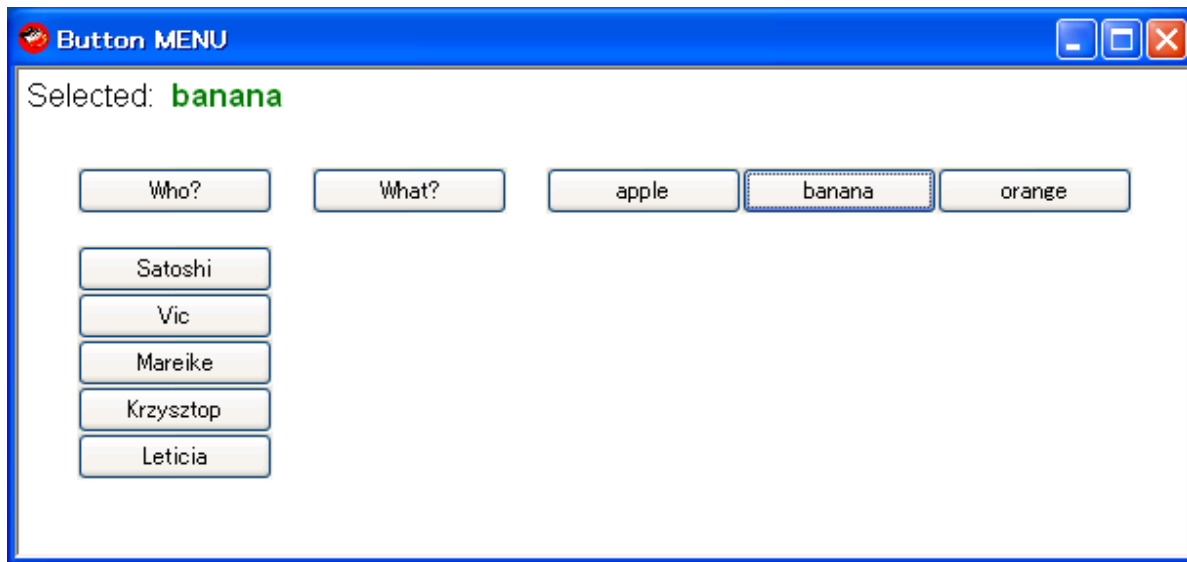
  para 'Selected: '
  @msg = para '', :stroke => green

  @toggle = true, true
  @f = []

  flow :left => 30, :top => 50, :width => 100 do
    menu 'who?', %w(Satoshi Vic Mareike Krzysztof Leticia), 0
  end
  @f << flow(:left => 30, :top => 90, :width => 100)

  flow :left => 150, :top => 50, :width => 100 do
    menu 'what?', %w(apple banana orange), 1
  end
  @f << flow(:left => 270, :top => 50, :width => 400)
end
```

sample33.png



And this one is a combination of sample32 and 33.

```

# sample34.rb
Shoes.app :title => 'Image MENU', :height => 250 do
  background lightskyblue.to_s..lightsalmon.to_s, :angle => 30

  def menu title, items, n
    nostroke
    nofill
    tb = image(:left => 0, :top => 0, :width => 100, :height => 21){rect
    para strong title
    @f ||= []
    @f << flow do
      items.each_with_index do |e, i|
        nostroke
        nofill
        b = image(:width => 100, :height => 21){rect(0, 0, 100, 21)}
        f = image(:width => 100, :height => 21){rect(0, 0, 100, 21, :fil
        yield b, f, i, e
        b.hover{f.show}
        b.leave{f.hide}
        b.click{@msg[n].text = strong(e)}
      end
    end.hide

    tb.click{@f[n].toggle; @msg[n].text = ''}
  end

  @msg = []
  para 'Selected who?: '
  @msg << para('', :stroke => forestgreen)
  para 'Selected what?: ', :left => 300
  @msg << para('', :stroke => tomato)

  flow :left => 30, :top => 50, :width => 100 do
    menu 'who?', %w(Satoshi Vic Mareike Krzysztop Leticia), 0 do |b, f,
      b.move 0, i*23
      f.move 0, i*23
      para i, '. ', e, "\n"
    end
  end

  flow :left => 150, :top => 50, :width => 400 do
    menu 'what?', %w(apple banana orange), 1 do |b, f, i, e|
      b.move((i+1)*102, -32)
      f.move((i+1)*102, -32)
      para "#{i}. #{e}", :left => 150 + (i+1)*102, :top => 50
    end
  end
end
end
end

```

sample34.png



If you want to hide an item when the mouse clicks on it, make the following revisions.

Line No. 20

`b.click{@msg[n].text = strong(e)} ----> b.click{@msg[n].text = strong(e); @f[n].toggle}`

Line No. 24

`tb.click{@f[n].toggle; @msg[n].text = ""} ----> tb.click{@f[n].toggle}`

The edited code is sample34-1.rb and the screenshot is sample34-1.png.

The original idea for this menu-like user interface was provided by Krzysztop Wicher.

sample34-1.png



arc and cap

New arc and cap methods were released in the 970th build. See the following article:

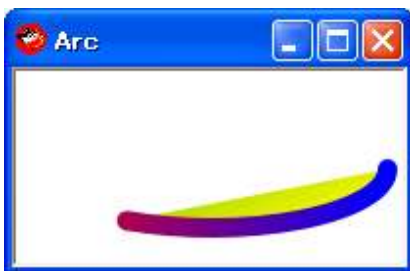
<http://newwws.shoooes.net/2008/09/10/arcs.html>

And `_why` shows us a wonderful combination with the `animate` method.

```
# sample35.rb
Shoes.app :width => 200, :height => 100, :title => 'Arc' do
  fill green.to_s..yellow.to_s, :angle => 45
  stroke red.to_s..blue.to_s, :angle => 90
  strokewidth 10
  cap :round

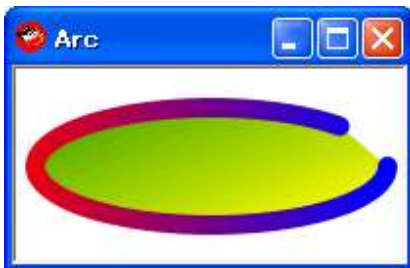
  a = animate 12 do |i|
    @c.remove if @c
    r = i * (PI * 0.01)
    @c = arc 100, 50, 180, 60, 0, i * (PI * 0.01)
    a.stop if r >= TWO_PI
  end
end
```

sample35.png



Started....

sample35-1.png



Almost finished....

widget with block

You can use the widget object with a block to respond to keypress or mouse events smoothly.

```
# sample36.rb
class Shoes::Creature < Shoes::Widget
  def initialize
    msg = para '', :stroke => white
    c = image '../images/yar.png'
    yield c, msg
  end
end

Shoes.app :width => 140, :height => 70 do
  flow :left => 10, :top => 10 do
    background blue.to_s..green.to_s, :width => 100, :height => 30
    creature do |c, msg|
      c.click do
        msg.text = 'Uhhhh...'
        a = animate(20){|i| c.rotate(-15); a.stop if i > 22}
      end
      c.hover{msg.text = 'hello'}
      c.leave{msg.text = ''}
    end
  end
end
```

sample36.png



Click Yar and she will rotate (*1).

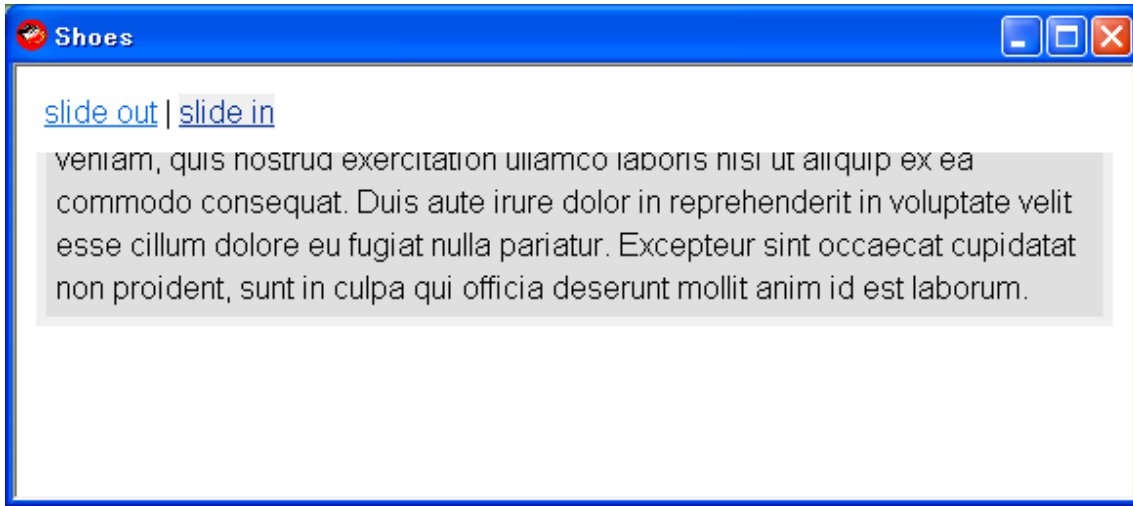
*1: With Shoes-0.r925, Yar rotates well as expected. But with Shoes-0.r970, Yar rotates when the mouse moves out of the Shoes window. This behavior is a bug. It has been fixed in Shoes-0.r1057.

Oops, Shoes-0.r1091 behaves as same as Shoes-0.r970. Maybe it's a bug again...

text message slide-in

_why gave us his one thousandth commit of Shoes on 24th Sep. Here is a new sample - simple-slide.rb: showing slide-in slide-out animation.

simple-slide.png



Next sample code, sample37.rb, which works almost similar behavior of the text message slide-in by using mask and animate methods.

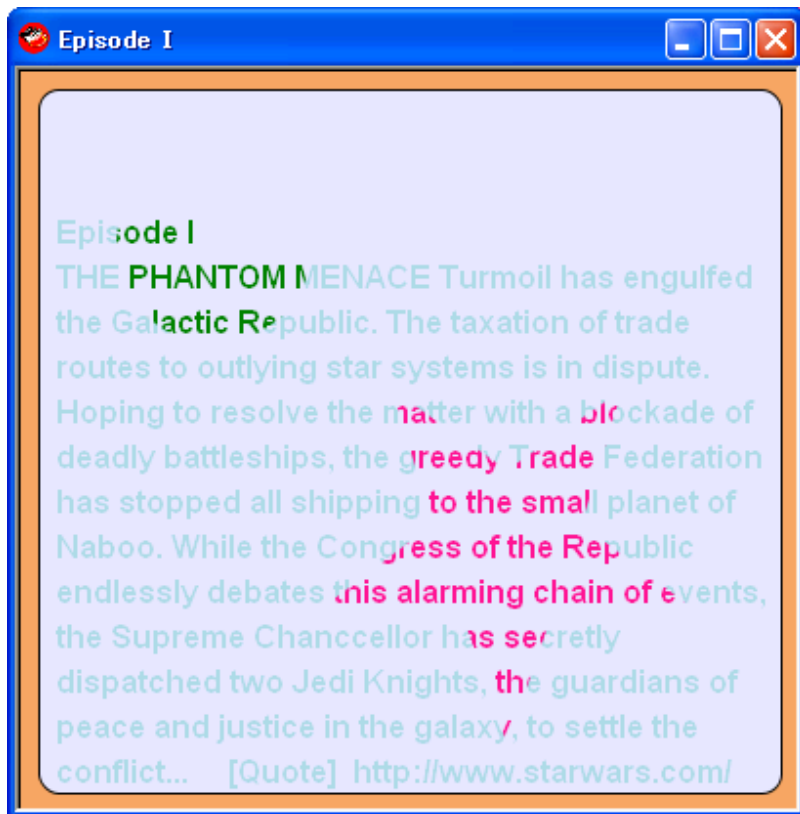
```
# sample37.rb

episode1 =<<-EOS
Episode I
THE PHANTOM MENACE Turmoil has engulfed the Galactic Republic. The taxat
Hoping to resolve the matter with a blockade of deadly battleships, the
while the Congress of the Republic endlessly debates this alarming chain
[Quote] http://www.starwars.com/episode-iii/bts/production/f2005012
EOS

Shoes.app :width => 400, :height => 380, :title => 'Episode I' do
  rect 0, 0, 400, 380, :fill => sandybrown
  rect 10, 10, 380, 360, :fill => lavender, :curve => 10
  stack do
    nostroke
    rect 10, 10, 380, 360, :fill => lightblue
    oval 50, 40, 100, :fill => green
    star 250, 245, 5, 100, 40, :fill => deeppink, :angle => 90
    mask do
      @t = para strong(episode1), :left => 15, :top => 340, :width => 38
    end
    @a = animate(36) do |i|
      @t.left, @t.top = 15, 340 - i
      @a.stop if i > 330
    end
  end
end
```

This is the screenshot.

sample37.png



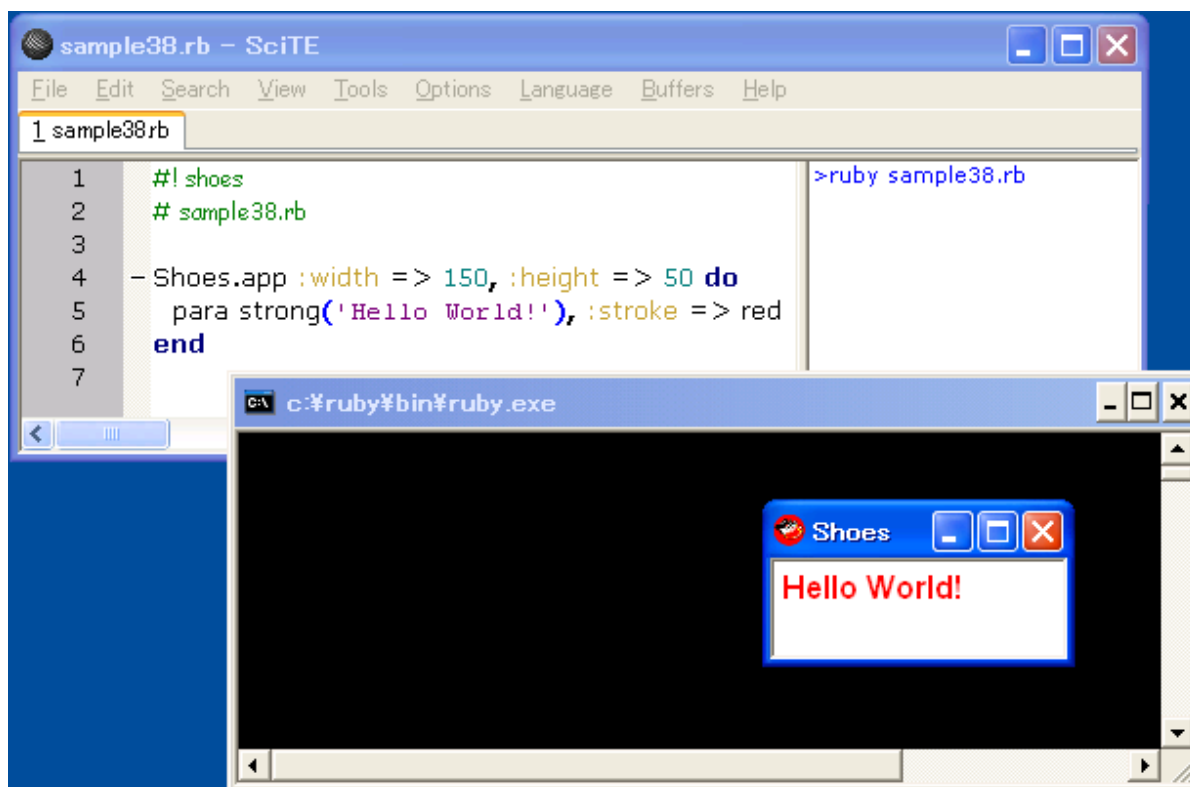
#! shoes

Sample38.rb has `#! shoes` on its first line. The shell will see that the program file has a `#!` line and pass it to Shoes.

```
#! shoes
# sample38.rb

Shoes.app :width => 150, :height => 50 do
  para strong('Hello world!'), :stroke => red
end
```

sample38.png



Write code with SciTE and push F5, then kick up Shoes!

And next. Sample38-1.rb is a Ruby program, not Shoes app, but it'll launch the Shoes app.

```
# sample38-1.rb

%x(ruby sample38.rb)
```

loading widgets from other files?

Sample39.rb has a require method to load the custom widget class stored in the other file (sample39-creature.rb).

loading widgets from other files?

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg01971.html>

The Main App and Its Requires

<http://help.shoooes.net/Rules.html>

```

# sample39.rb

require 'sample39-creature'

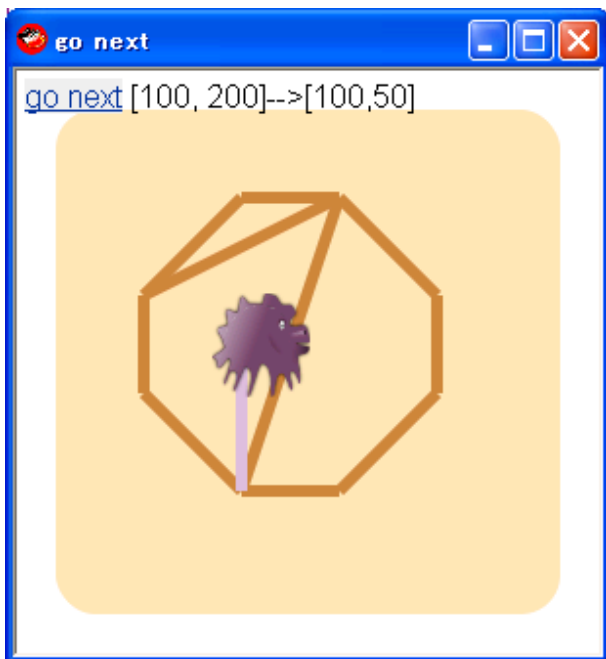
Shoes.app :title => 'go next', :width => 300, :height => 300 do
  background moccasin, :margin=> 20, :curve => 20
  c = creature('../images/loogink.png', 50, 100)

  routes = [[100, 50], [150, 50], [200, 100], [200, 150], [150, 200], [100, 200], [50, 100], [50, 50], [100, 50]]
  i = -1
  para link('go next'){
    begin
      x, y = routes[(i+=1) % 10]
      @msg.text = "#{c.position.inspect}-->[#{x},#{y}]"
      c.glide [x, y], :line => true
    end unless c.playing?
  }

  @msg = para ''
end

```

sample39.png



```

# sample39-creature.rb
class Shoes::Creature < Shoes::Widget
  def initialize path, x, y
    @path = path
    @img = image path
    @img.move x, y
  end

  def glide args, opt = {:line => false}
    args << @img.left << @img.top
    x1, y1, x0, y0 = args.collect{|e| e.to_f}

    a = animate(48) do |i|
      @playing = true
      case
      when x0 < x1
        x = x0 + i
        y = y0 + (y1 - y0) / (x1 - x0) * i if y0 < y1
        y = y0 if y0 == y1
        y = y0 - (y0 - y1) / (x1 - x0) * i if y0 > y1
        max = x1 - x0
      when x0 == x1
        x = x0
        y = y0 + i if y0 < y1
        y = y0 - i if y0 > y1
        y = y0 if y0 == y1
        max = (y1 - y0).abs
      when x0 > x1
        x = x0 - i
        y = y0 + (y1 - y0) / (x0 - x1) * i if y0 < y1
        y = y0 if y0 == y1
        y = y0 - (y0 - y1) / (x0 - x1) * i if y0 > y1
        max = x0 - x1
      else
      end

      @l.remove if @l
      strokewidth 6
      @l = line(x0 + 15, y0 + 15, x.to_i + 15, y.to_i + 15, :stroke => t
      #@img.move x.to_i, y.to_i
      @img.remove
      @img = image @path, :left => x.to_i, :top => y.to_i

      if i == max
        a.stop
        @playing = false
        line(x0 + 15, y0 + 15, x.to_i + 15, y.to_i + 15, :stroke => peru
        @img.remove
        @img = image @path, :left => x.to_i, :top => y.to_i
      end
    end
  end

  def position
    [@img.left, @img.top]
  end

  def playing?
    @playing
  end
end

```


optional arguments

When we create an oval shape, like

```
oval :left => 50, :top => 50, :width => 30
```

The :left and :top positions are the top-left corner of the oval.

But we create an star shape, like

```
star :left => 50, :top => 50, :points => 5, :outer => 15, :inner => 10
```

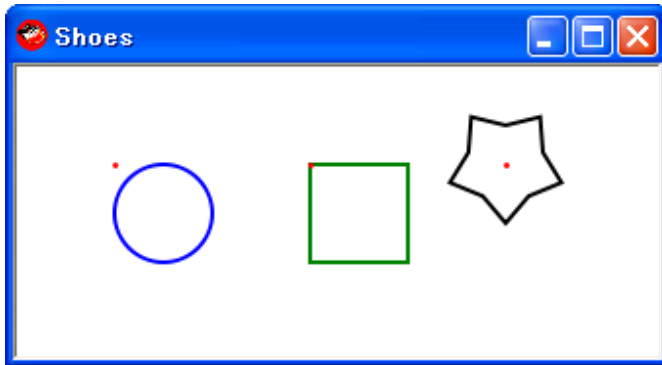
The :left and :top positions are the center of the star.

A bit strange behavior...

This information was provided by Sergio Silva.

```
# sample40.rb
Shoes.app :width => 330, :height => 150 do
  nofill
  strokewidth 2
  oval 50, 50, 50, :stroke => blue
  rect 150, 50, 50, 50, :stroke => green
  star 250, 50, 5, 30, 20, :stroke => black
  oval 50, 50, 1, :stroke => red, :fill => red
  oval 150, 50, 1, :stroke => red, :fill => red
  oval 250, 50, 1, :stroke => red, :fill => red
end
```

sample40.png



We can use the :center option to specify the coordinates. But it works well only in the case of the oval and rect, not the star method. If we add an undefined option like :oops as one of the arguments, no error will occur and nothing will happen, it will just be ignored.

I don't know if this behavior is a spec or a bug...

```

# sample40-1.rb
Shoes.app do
  stack :width => 0.3 do
    nofill
    strokewidth 2
    oval 50, 50, 1, :stroke => red, :fill => red
    @o = oval 50, 50, 50, :stroke => blue, :center => false, :oops => tr
    @p1 = para '', :top => 150
  end

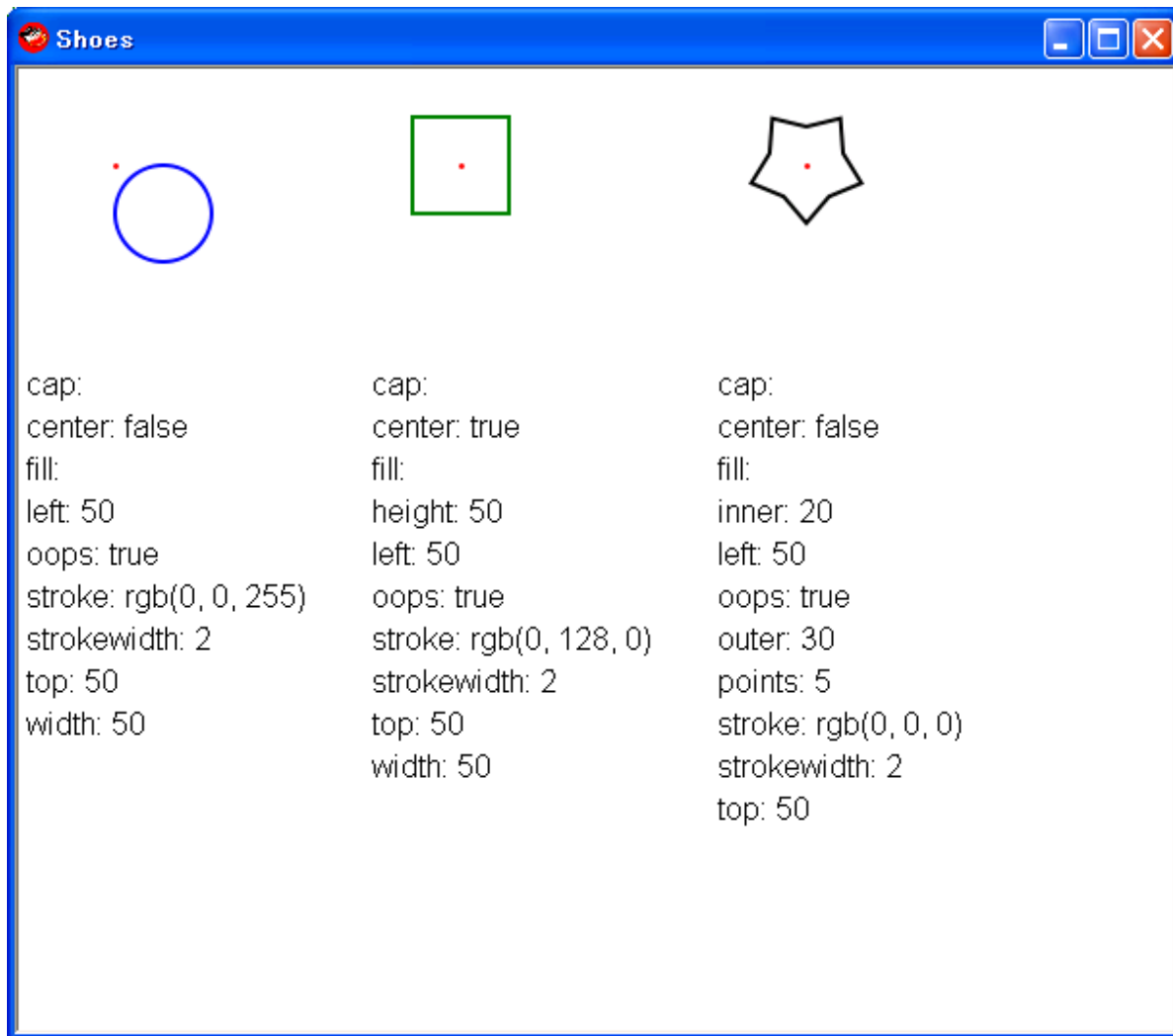
  stack :width => 0.3 do
    nofill
    strokewidth 2
    oval 50, 50, 1, :stroke => red, :fill => red
    @r = rect 50, 50, 50, 50, :stroke => green, :center => true, :oops =
    @p2 = para '', :top => 150
  end

  stack :width => 0.4 do
    nofill
    strokewidth 2
    oval 50, 50, 1, :stroke => red, :fill => red
    @s = star 50, 50, 5, 30, 20, :stroke => black, :center => false, :oo
    @p3 = para '', :top => 150
  end

  @p1.text = @o.style.map{|e| e.join(': ')}>.sort>.join("\n")
  @p2.text = @r.style.map{|e| e.join(': ')}>.sort>.join("\n")
  @p3.text = @s.style.map{|e| e.join(': ')}>.sort>.join("\n")
end

```

sample40-1.png



In the above sample40-1, the oval and rect methods accepted the `:center` option, but the star method ignored it as it ignored the undefined option `:oops`.

slot with scrollbar

The `:scroll` option establishes a slot as a scrolling slot.

```
# sample41.rb
Shoes.app :width => 240, :height => 161, :resizable => false do
  image '../images/jellybeans.jpg'
  flow :width => 100, :height => 40, :left => 2, :top => 2, :scroll => t
  background bisque
  30.times do |i|
    color = COLORS.keys.map{|sym|sym.to_s}.sort_by{rand}
    para "colorful jellybeans", :stroke => send(color.first)
  end
end
end
end
```

sample41.png



The :state style

The :state style is for disabling or locking certain controls if you do not want them to be edited.

```
# sample42.rb
Shoes.app :width => 400, :height => 410 do
  src = IO.read($PROGRAM_NAME)
  background deepskyblue

  stack do
    caption strong ":state >> string"
    para '# sample42.rb'
  end

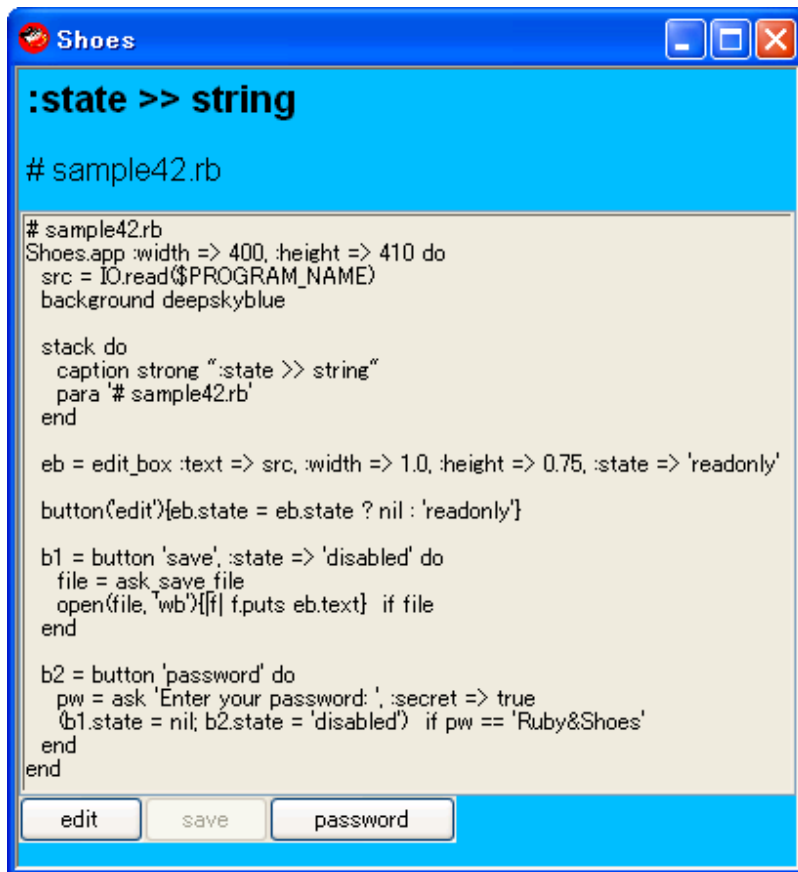
  eb = edit_box :text => src, :width => 1.0, :height => 0.75, :state =>

  button('edit'){eb.state = eb.state ? nil : 'readonly'}

  b1 = button 'save', :state => 'disabled' do
    file = ask_save_file
    open(file, 'wb'){|f| f.puts eb.text} if file
  end

  b2 = button 'password' do
    pw = ask 'Enter your password: ', :secret => true
    (b1.state = nil; b2.state = 'disabled') if pw == 'Ruby&Shoes'
  end
end
```

sample42.png



I had used ARGV[0] to get the file name. But it's not correct usage. We have to use \$0 or \$PROGRAM_NAME instead of ARGV[0].

See [this](#)

Shoes::FONTS and External Fonts

Shoes::FONTS is a complete list of the fonts you can use.
Loading from external fonts, such as TrueType and OTF files.

References are

New Today: External Fonts

<http://newwws.shoooes.net/2008/10/06/new-external-fonts.html>

Shoes Manual: font

<http://help.shoooes.net/Built-in.html#font>

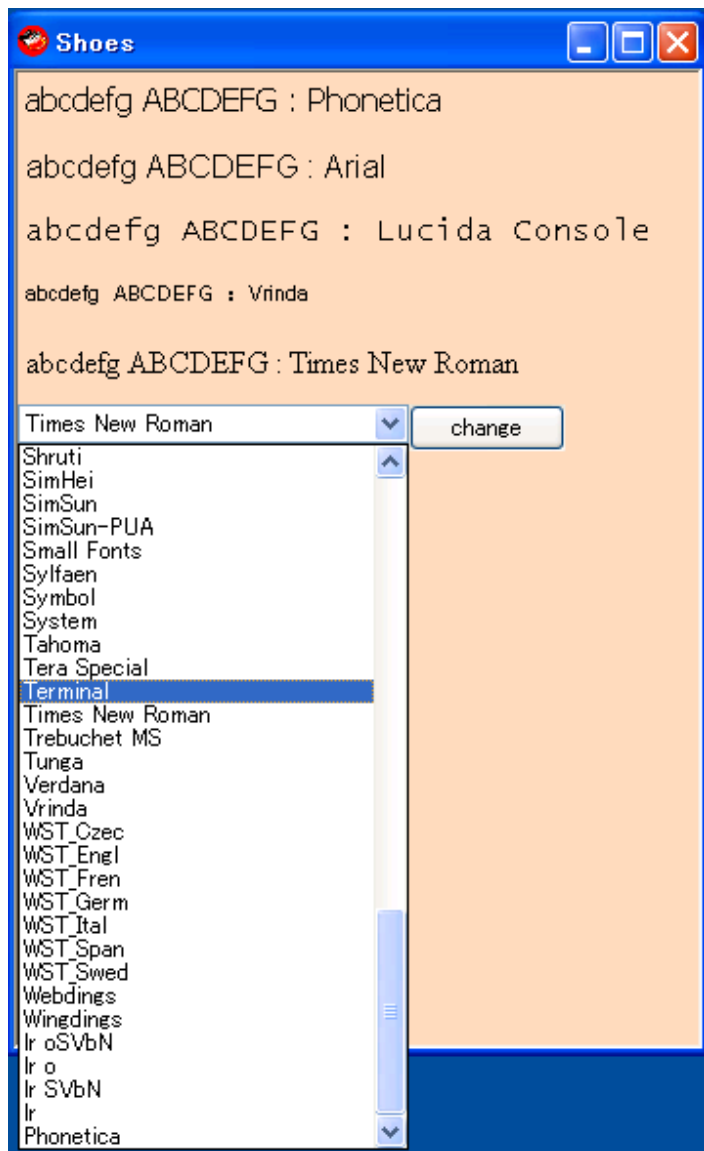
```

# sample43.rb
font "phonetica.ttf"

Shoes.app :width => 350, :height => 500 do
  background peachpuff
  font = 'Phonetica'
  slot = stack{para 'abcdefg ABCDEFG : ' + font, :font => font}
  font = list_box :items => (Shoes::FONTS << "Phonetica"), :height => 30
  button 'change' do
    slot.append{para 'abcdefg ABCDEFG : ' + font.text, :font => font.tex
  end
end

```

sample43.png



Shoes Tutorial Note Launcher

Markdown + BlueCloth + Shoes = AWESOME!

Markdown

<http://daringfireball.net/projects/markdown/>

BlueCloth

<http://www.deveiate.org/projects/BlueCloth>

```

# sample44.rb
Shoes.setup do
  gem 'BlueCloth'
end

require 'BlueCloth'
BROWSER = 'C:/Program Files/Mozilla Firefox/firefox.exe'
PROTOCOL = 'file:/'
mfolder = File.dirname(Dir.pwd) + '/mdowns'
hfolder = File.dirname(Dir.pwd) + '/html'

Shoes.app :width => 450, :height => 130, :title => 'Shoes Tutorial Note
  background dimgray..gainsboro, :angle => 90
  @slot = stack{}

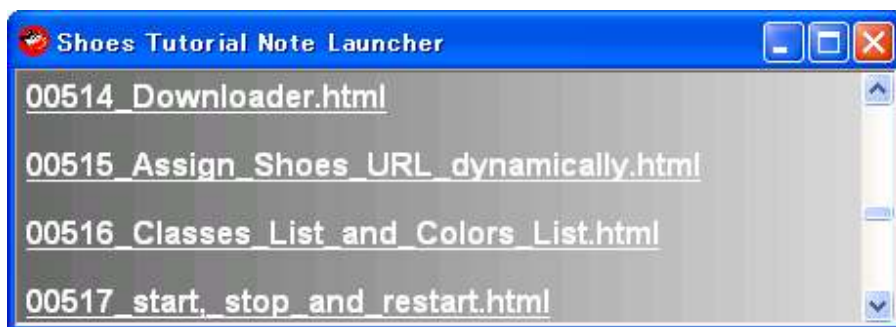
  Dir.entries(mfolder).each do |mname|
    @slot.append do
      hname = mname.sub(/.mdown/, '.html')
      mfile = mfolder + '/' + mname
      hfile = hfolder + '/' + hname
      para link(strong(hname), :stroke => white){
        b = BlueCloth.new IO.read(mfile)
        open(hfile, 'w'){|f| f.puts b.to_html}
        system BROWSER, PROTOCOL + hfile
      } if /.mdown$/ =~ mname
    end
  end
end

```

Note:

BlueCloth has a bug. It may delete \ in the code falsely. :(
So, please use sample44.rb under src directory instead of above code.

sample44.png



UTF-8

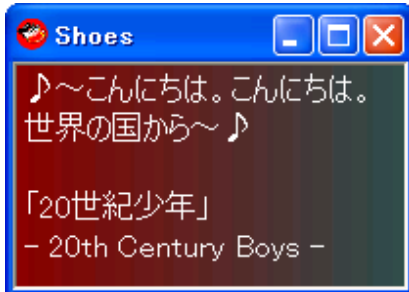
Shoes expects all strings to be in UTF-8 format.

UTF-8 Everywhere

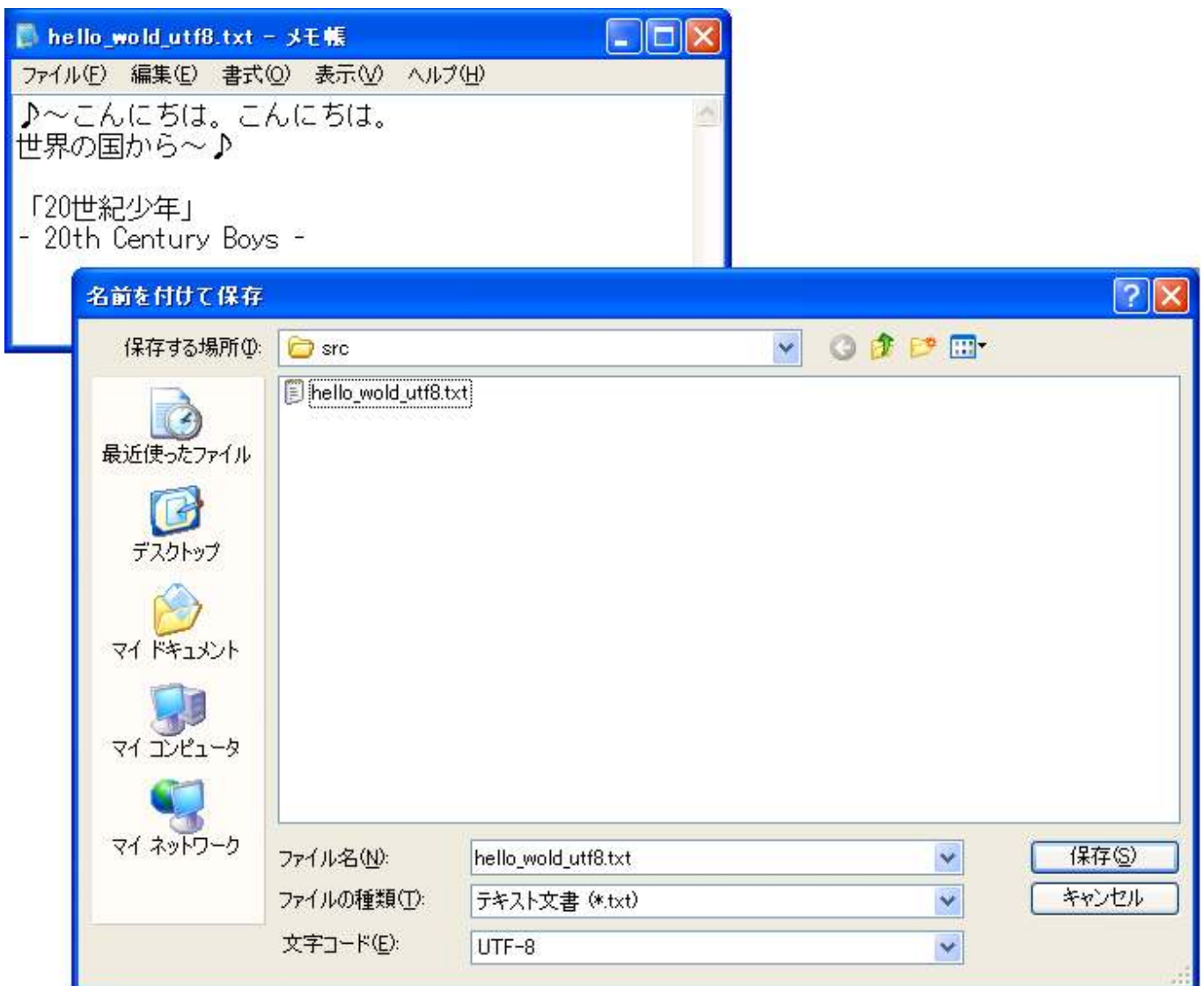
<http://help.shoooes.net/Rules.html>

```
# sample45.rb
Shoes.app :width => 200, :height => 115 do
  background darkred..darkslategray, :angle => 90
  para IO.read('hello_wold_utf8.txt'), :font => "MS UI Gothic", :stroke
end
```

sample45.png



sample45-1.png



This Japanese text editor uses UTF-8.

Open a new app window

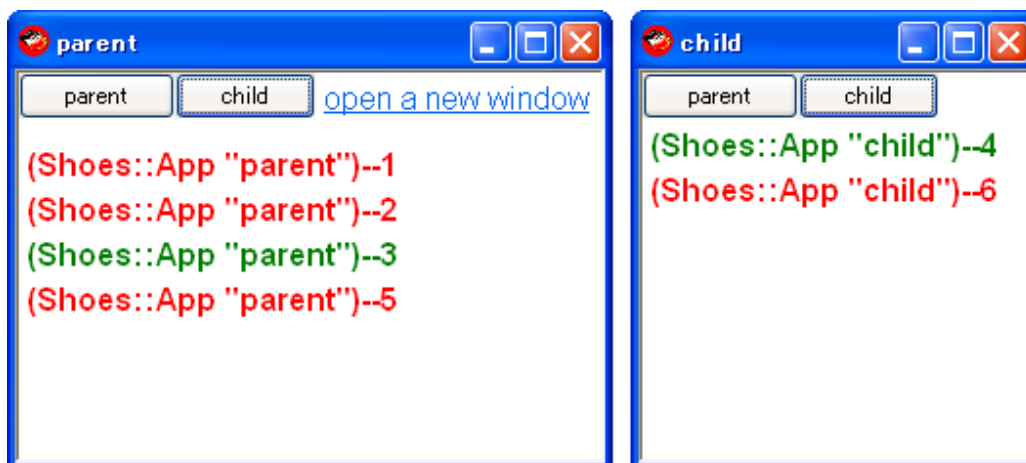
We can use window method to open a new app window.

```
# sample46.rb
Shoes.app :title => 'parent', :width => 300, :height => 200 do
  def open_new_window
    window :title => 'child', :width => 200, :height => 200 do
      button('parent'){owner.hello green}
      button('child'){owner.hello green, self}
    end
  end

  def hello color, win = nil
    win ||= self
    @n ||= 0
    @n += 1
    win.para strong("#{win}--#{@n}\n"), :stroke => color
  end

  button('parent'){hello red}
  button('child'){hello red, @w}
  para link('open a new window'){@w = open_new_window}
end
```

sample46.png



This screenshot shows the following.

- open parent window.
- in parent window, click parent button, output is --1
- in parent window, click child button, output is --2
- click 'open a new window' link, open a child window.
- in child window, click parent button, output is --3
- in child window, click child button, output is --4
- in parent window, click parent button, output is --5
- in parent window, click child button, output is --6

Another example.

We can use Shoes.app method to open a new app window.

```

# sample48.rb
@blk = class Trip < Shoes
  url "/", :index
  url "/japan", :japan
  url "/india", :india
  url "/tokyo", :tokyo
  url "/pune", :pune

  @@win = 0

  def index
    case @@win
    when 0
      background coral
      para strong link("Go to Japan.", :click => "/japan")
      para strong link("Go to India.", :click => "/india")
    when 1
      background crimson
      para strong link("Go to Tokyo.", :click => "/tokyo")
    when 2
      background darkorange
      para strong link("Go to Pune.", :click => "/pune")
    else
    end
  end
end

def japan
  @@win = 1
  Shoes.app :title => "Japan", :width => 200, :height => 100, &@blk
  @@win = 0
  visit "/"
end

def india
  @@win = 2
  Shoes.app :title => "India", :width => 200, :height => 100, &@blk
  @@win = 0
  visit "/"
end

def tokyo
  background gold
  para strong "welcome to Tokyo!"
end

def pune
  background darksalmon
  para strong "welcome to Pune!"
end
end

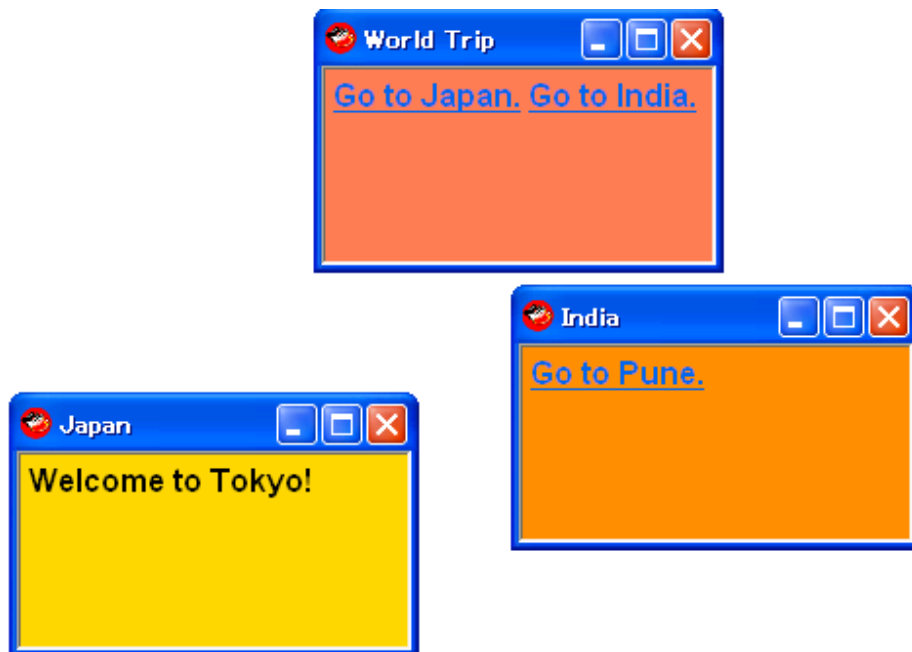
Shoes.app :title => "world Trip", :width => 200, :height => 100, &@blk

```

Note:

Code Highlighter has a bug. It may replace &@ to &:@ in the code falsely. :(
So, please use sample48.rb under src directory instead of above code.

sample48.png



This screenshot shows the following.

- open first window: title is World Trip
- in first window, click 'Go to Japan', open second window: title is Japan
- in first window, click 'Go to India', open third window: title is India
- in second window, click 'Go to Tokyo', change Shoes-URL on the same window, then shows the message "Welcome to Tokyo!"

The original idea was discussed in the Shoes ML.

[links in ur windoze](#)

Open the Shoes console window from your app

A little snippet to open the Shoes console window from your app.

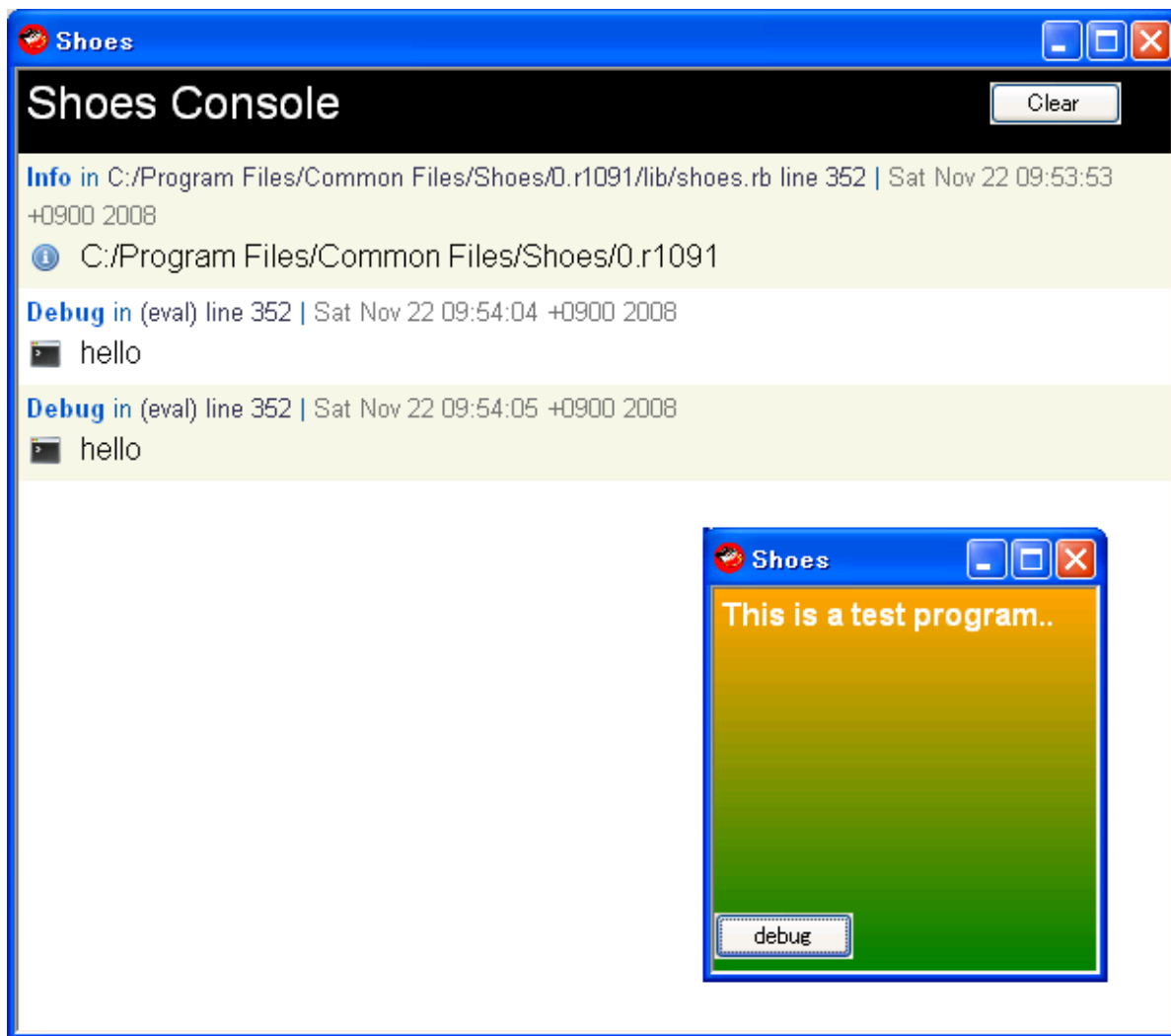
```
# sample51.rb
require File.join(DIR, 'lib/shoes/log')

Shoes.app :width => 200, :height => 200 do
  window{extend Shoes::LogWindow; setup}

  # write your app code below
  background orange..green
  para strong 'This is a test program..', :stroke => white

  info DIR
  button 'debug', :bottom => 0, :left => 0 do
    debug 'hello'
  end
end
```

sample51.png



This screenshot shows the following.

- click the button 'debug' twice on your app window

Another way from the Shoes ML.

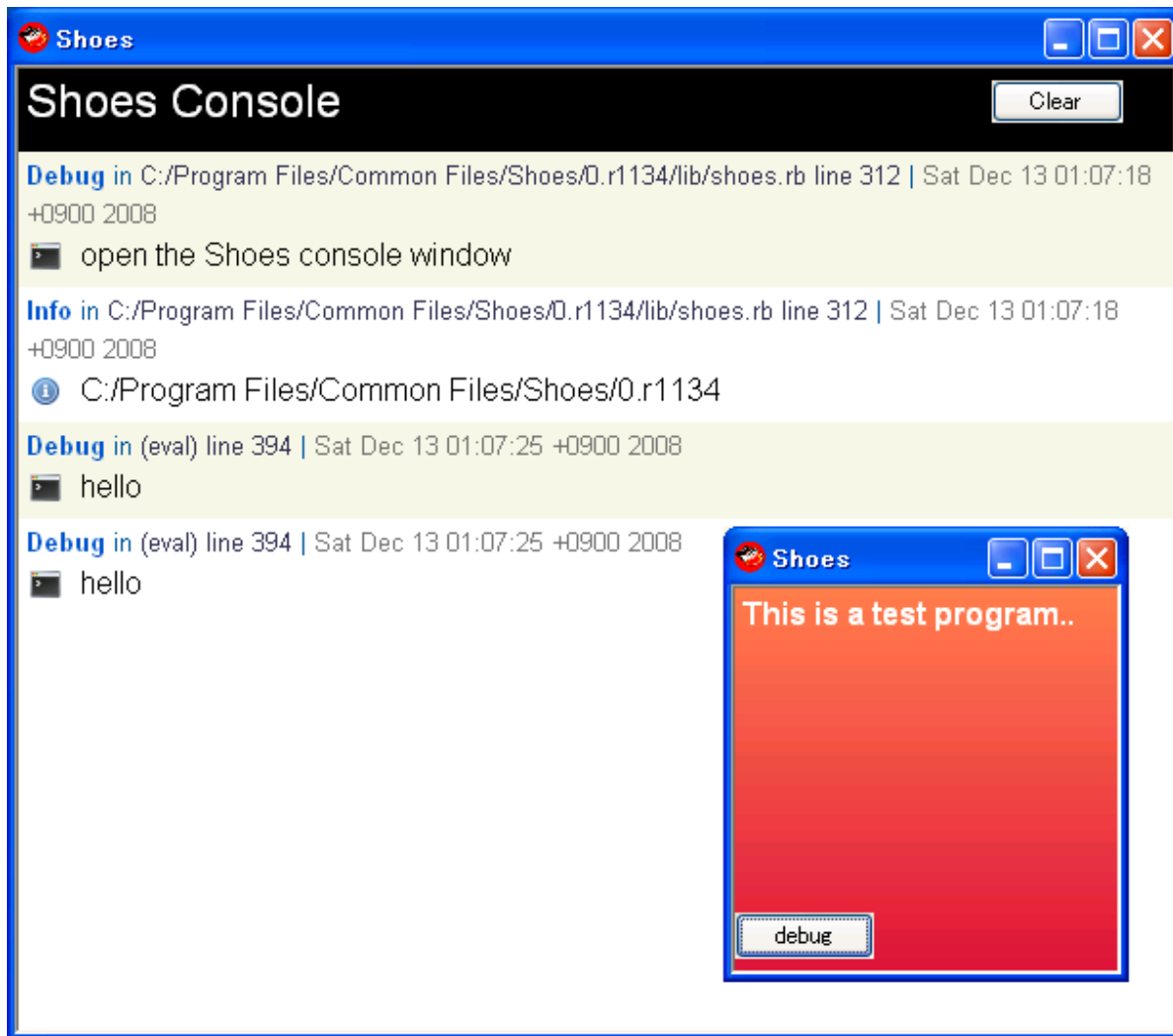
<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg02676.html>

```
# sample55.rb
Shoes.app :width => 200, :height => 200 do
  Shoes.show_log
  debug 'open the shoes console window'

  # write your app code below
  background coral..crimson
  para strong 'This is a test program..', :stroke => white

  info DIR
  button 'debug', :bottom => 0, :left => 0 do
    debug 'hello'
  end
end
```

sample55.png



Customize Shoes Class

You know,... creating Shoes app is writing Ruby code.

Hence, we can customize Shoes Class with Ruby overwriting and overloading feature.

This is no wonder, but I just noticed. :-P

```

# sample53.rb
class Shoes::Image
  def small
    self.style :width => self.width / 2, :height => self.height / 2
  end

  def big
    self.style :width => self.width * 2, :height => self.height * 2
  end
end

PATH = '../images/yar.png'

Shoes.app :width => 250, :height => 150 do
  w, h = imagesize(PATH)
  img = image PATH, :width => w, :height => h, :name => PATH.split('/').
  msg = para 'ready', :left => w, :top => h
  every 3 do
    img.style[:width] > w ? img.small : img.big
    msg.text = "#{img.style[:name]} width is : #{img.style[:width]}" + "
               "#{img.style[:name]} height is : #{img.style[:height]}"
  end
end

```

sample53.png



Image Effects with blur method

Shoes 2 has some new features. I have attempted to use blur method to make images rise up gradually.

INSIDE SHOES 2

<http://shoooes.net/about/raisins/>

```

#sample54.rb
Shoes.app :width => 150, :height => 150 do
  def blur_creature img
    a = animate 6 do |i|
      name, top, left = img.style[:name], img.style[:top], img.style[:left]
      img.remove
      img = image name, :name => name, :top => top, :left => left
      img.blur 20 - i
      img.show
      a.stop if i > 20
    end
  end

  click do
    clear do
      2.times do |i|
        name = "../images/#{%w[loogink cy kamome shaha yar][rand(5)]}.png"
        blur_creature image(name, :name => name, :top => 20 + i * 60, :left => i * 60)
      end
    end
  end
end
end

```

sample54.png



Note

Shoes may crash without very enough interval between mouse clicks. Oh,... :(

Video playback

We can do playback YouTube videos on Shoes!

I referred to the following web site.

[Gulfy your Ruby apps with Shoes](#)

```

# sample59.rb
URL = 'http://jp.youtube.com/watch?v=8hBrRZuXjHA'

Shoes.app :width => 420, :height => 330, :title => 'YouTube Viewer v0.1'
def youtube
  style Inscription, :stroke => white, :weight => 'bold'

  r1 = rect :left => 10, :top => 310, :width => 30, :height => 15
  r2 = rect :left => 45, :top => 310, :width => 40, :height => 15
  r3 = rect :left => 90, :top => 310, :width => 30, :height => 15
  r4 = rect :left => 140, :top => 310, :width => 20, :height => 15

  background orange..lime, :angle => 90
  @msg = inscription '', :left => 170, :top => 305, :stroke => darkred

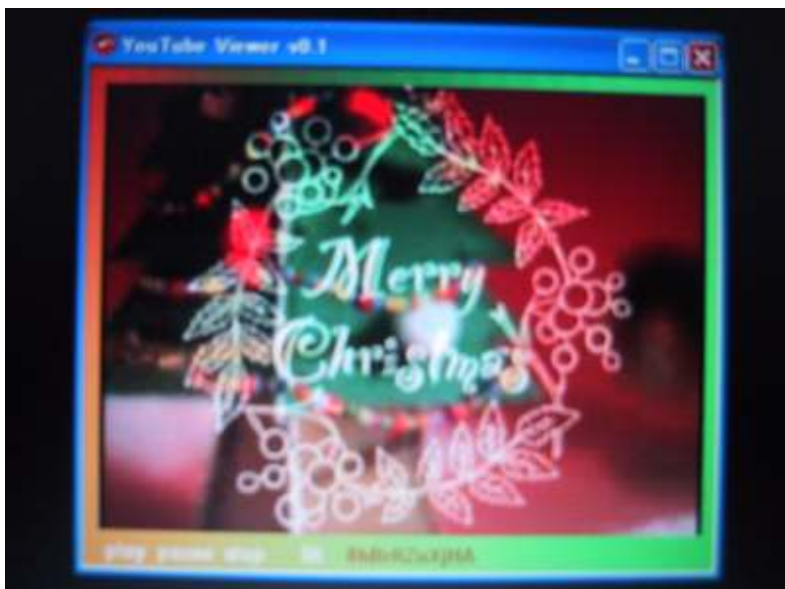
  url = ask 'URL: '
  url = URL unless url
  base, vid = url.split 'watch?v='
  @msg.text = vid
  download "#{base}watch?v=#{vid}" do |page|
    t = /, "t": "([^"]+)"/.match(page.response.body)[1]
    @v = video("#{base}get_video?video_id=#{vid}&t=#{t}", :autoplay =>
  end

  inscription 'play', :left => 10, :top => 305; r1.click{@v.play}
  inscription 'pause', :left => 45, :top => 305; r2.click{@v.pause}
  inscription 'stop', :left => 90, :top => 305; r3.click{@v.stop}
  inscription 'DL', :left => 140, :top => 305; r4.click{@v.stop; @v.r
end

youtube
end

```

sample59.png




```
# sample59-1.rb
Shoes.app(:width => 400, :height => 300){video \
"http://jp.youtube.com/get_video?video_id=8hBrRZuxjHA&t=\
OEgSToPDskKhuxfknhF5eWP0vhe-nw5P", :autoplay => true}
```

Just one-liner solution playing YouTube video on Shoes!

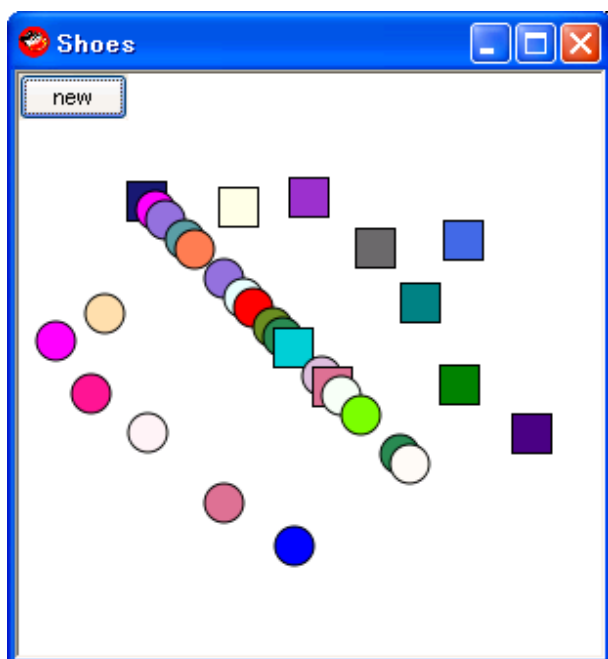
Scope: local variable and instance variable

We need to learn the scope, the difference between local variable and instance variable, over and over again.

```
# sample60.rb
Shoes.app :width => 300, :height => 300 do
  i = 45
  button 'new' do
    i += 5
    box = rand(2) == 0 ? rect(i, i, 20) : oval(i, i, 20)
    box.style :fill => send(COLORS.keys.map{|sym| sym.to_s}[rand(COLORS.k

    @flag = false
    box.click{@flag = true; @box = box}
    box.release{@flag = false}
    motion{|left, top| @box.move(left-10, top-10) if @flag}
  end
end
```

sample60.png



This snapshot is created by the following setps.

- clicked 'new' button 30 times
- picked up and drew 6 ovals to the lower side
- picked up and drew 7 rects to the upper side

edit_line with block

By using edit_line with block, this app can just update the sum every time one of the values gets changed. :-D

Original code was created by Michael Kohl
See the following his repos.

[citizen428 / littlesteps / little_helper.rb](#)

[the Little Helper in the Shoebox](#)

```
#sample63.rb
Shoes.app :title => "Little Helper v0.2", :width => 210, :height => 325

@money = para '', :top => 250, :left => 10

def calculate h = @hours.text.to_f, w = @wage.text.to_f, t = @tax.text
  @money.text = h * w * ((100 - t) / 100.0) * 4.3
end

stack :margin => 10 do
  caption "Awful numbers"

  para "Hours/week "
  @hours = edit_line(15, :width => 50){calculate}

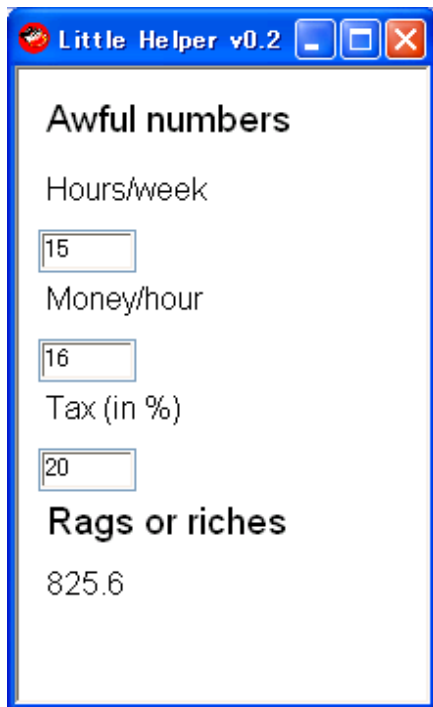
  para "Money/hour"
  @wage = edit_line(16, :width => 50){calculate}

  para "Tax (in %)"
  @tax = edit_line(20, :width => 50){calculate}

  caption "Rags or riches"
end

calculate 15, 16, 20
end
```

sample63.png



One way of layer manipulation

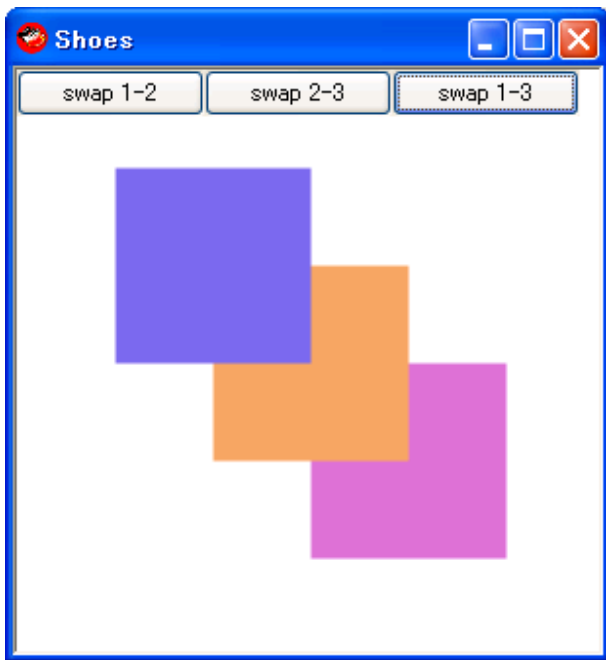
Usually the later-made Shape instances are layouted on the early-made ones. So, attempted to write a code to swap the layers between them.

```
# sample64.rb
Shoes.app :width => 300, :height => 300 do
  def swap s1, s2
    tmp = s2.style.to_a
    s1.style.to_a.each{|k, v| s2.style k => v}
    tmp.each{|k, v| s1.style k => v}
  end

  nostroke
  r1 = rect 50, 50, 100, 100, :fill => mediumslateblue
  r2 = rect 100, 100, 100, 100, :fill => sandybrown
  r3 = rect 150, 150, 100, 100, :fill => orchid

  button('swap 1-2'){swap r1, r2}
  button('swap 2-3'){swap r2, r3}
  button('swap 1-3'){swap r1, r3}
end
```

sample64.png



Show and hide the slots

At first, preparing all slots hidden in piles. Then do show or hide them at the right moment. It works as like using `url` method.

Original idea was given by [Michael Kohl](#).

```
# sample66.rb
Shoes.app :width => 200, :height => 200 do
  background tan
  st = {:left => 10, :top => 10, :width => 180, :height => 180}

  slots = []
  10.times do |i|
    slots << stack do
      color = COLORS.keys[rand(COLORS.keys.length)]
      background COLORS[color], :curve => 20
      para i
      para color.to_s
    end.hide
  end

  slots.each{|s| s.style st}

  every 1 do |i|
    hit = slots[i%10]
    slots.each{|s| s == hit ? s.show : s.hide}
  end
end
```

sample66.png



class definition outside of Shoes.app block

If you want to define **class** outside of Shoes.app block, attention of **self**. Please run and read the following snippets.

This tip was created in the discussion with Paul Harris on the Shoes course 3rd batch.

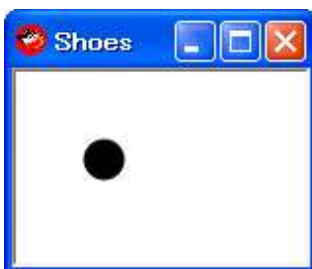
```
# sample69.rb
class Sheep
  attr_writer :avatar

  def draw x, y
    @avatar.move x, y
  end
end

Shoes.app :width => 150, :height => 100 do
  s = Sheep.new
  s.avatar = oval 0, 0, 20, 20

  a = animate do |i|
    s.draw i, i
    a.stop if i > 50
  end
end
```

sample69.png



```
# sample69-1.rb
class Sheep
  def initialize
    @avatar = $app.oval 0, 0, 20, 20
  end

  attr_writer :avatar

  def draw x, y
    @avatar.move x, y
  end
end

Shoes.app :width => 150, :height => 100 do
  $app = self
  s = Sheep.new

  a = animate do |i|
    s.draw i, i
    a.stop if i > 50
  end
end
```

sample69-1.png Another way of recreating sample69.

```
# sample69-2.rb
class Sheep < Widget
  def initialize
    oval 0, 0, 20, 20
  end

  def run x, y
    move x, y
  end
end

Shoes.app :width => 150, :height => 100 do
  s = sheep

  a = animate do |i|
    s.run i, i
    a.stop if i > 50
  end
end
```

sample69-2.png Another way of recreating sample69.

Note

In above sample69-2.rb, I use the method name run instead of draw. Because Shoes has already defined draw method, hence if I use the same name, it is overloaded and Shoes can't open app window.

Hot Topics in the Shoes ML and Shoooes.net

Picked up some topics here which were discussed in the Shoes ML nowadays.

External Fonts

_why added support to Shoes for loading .ttf and .otf files (and others, depending on your platform.)

Can't wait next build.

external font files

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg02092.html>

Locking edit_box

If Shoes makes the edit_box read-only, we can select (copy) text data from it.

Locking edit boxes

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg02120.html>

Styling Master List

It's the last big missing piece of the built-in manual.

The Styles Master List

<http://help.shoooes.net/Styles.html>

Trying to ease the RubyGems pain

_why announced that he is trying to ease the RubyGems pain.

Some issues are:

- RubyGems doesn't have a GUI.
- Shoes users shouldn't be expected to use the commandline. (So, no `gem install twitter`.)
- Shoes users shouldn't need admin rights to use a lib.
- Shoes needs to include SQLite3, for the image cache.
- And I like having Hpricot and JSON.

The hpricot, sqlite3, json gems

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg02295.html>

Shoes snapshot

Shoes snapshot feature will be coming soon!

_why was in favor of our wish. :)

Wish for screenshot feature

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg02781.html>

shoes gem alternate repo's

Here is the way to install a gem from gems.github.com.

[shoes gem alternate repo's](#)

```
#-----
Shoes.setup do
  Gem.sources = %w[http://gems.github.com/ http://gems.rubyforge.org/]
  gem 'dxoigmn-pcaprub'
end

Shoes.app do
  require 'pcaprub'
end

#-----
Shoes.app do
  source "http://gems.github.com";
  gem "foo-bar"
end

#-----
Shoes.setup do
  gem "something-from-rubyforge"
  gem "more-rubyforge"
  gem "anything-will-come-from-rubyforge-by-default"
  source "http://gems.github.com";
  gem "now-pulling-from-github"
  gem "etc-and-so-forth"
end
```

http post in shoes

Use the download method, which can do HTTP in the background and uses platform-specific code to work much, much faster. by _why

[help with http post in shoes](#)

```
#-----
Shoes.app do
  download "http://localhost:8080", :method => "POST",
    :body => "userName=user&password=11111" do |dl|
    para dl.response.body
  end
end
```

Exercises

Exercise 1 twitter client (reader)

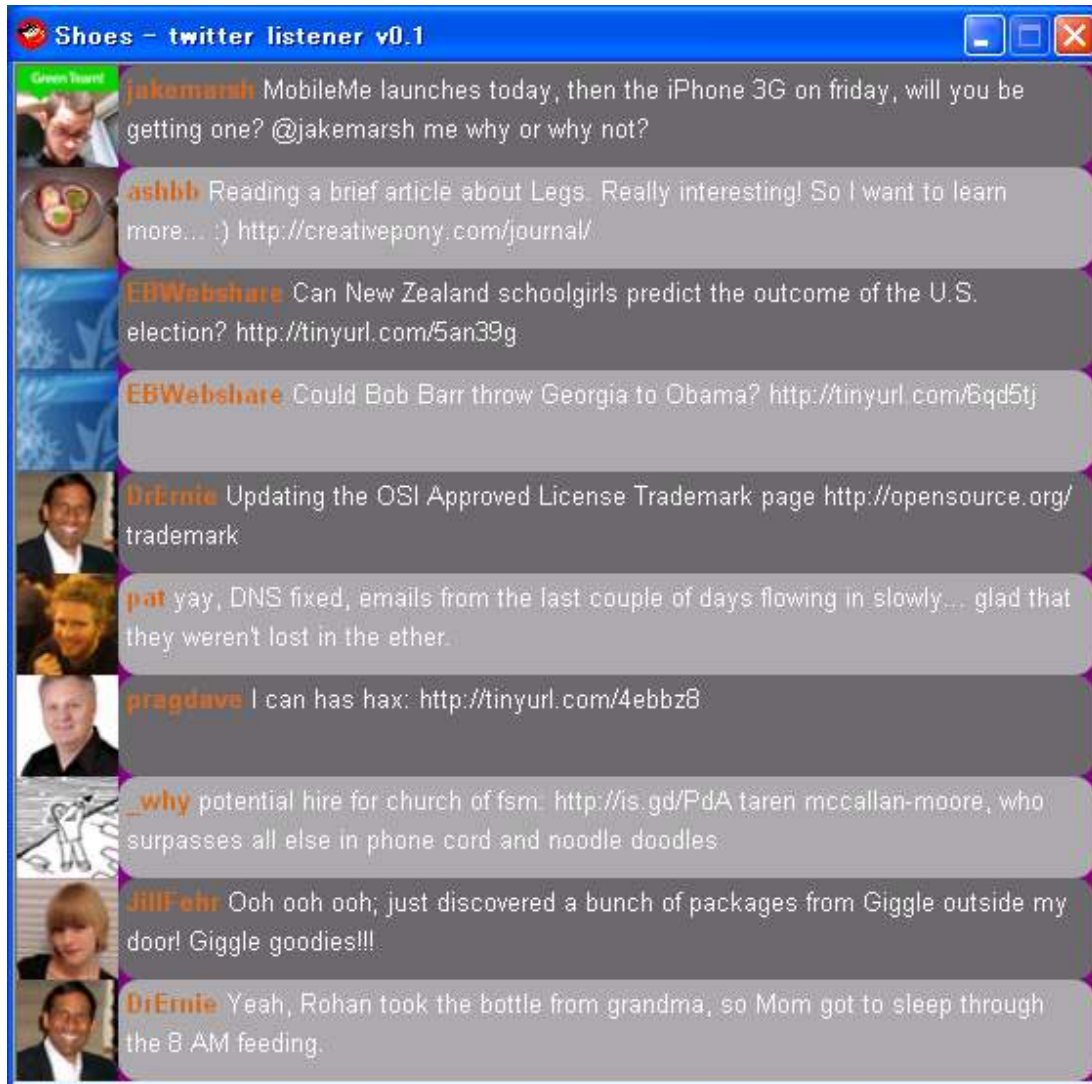
The following spec is an example.

Let's imagine freely and write your own twitter listener.

Example spec:

1. Access your twitter homepage: <http://twitter.com/home>
2. Get the friends timeline: [/statuses/friends_timeline.xml](#)
3. Display the latest 10 twitters.
4. User interface image is:

twitter_listener_snapshot.png



Have fun!

Exercise 2 footracer

The following spec is an example.

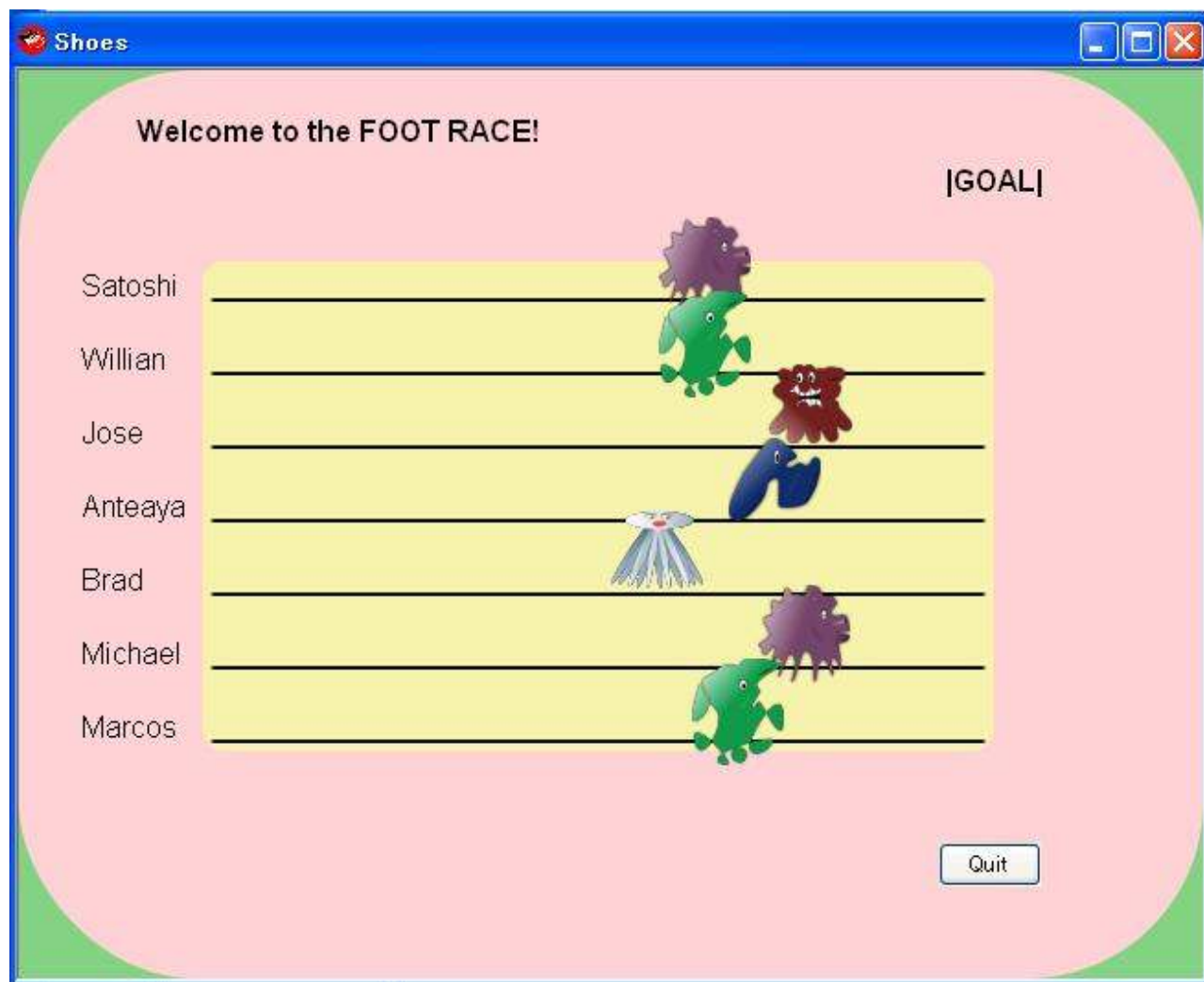
Let's imagine freely and write your own Foot Race Game.

Example spec:

1. Racers run toward the goal. When the first racer meets the goal line, the game stops and then shows the winner.

2. When multiple racers meet the goal line at a time, they are all winners.
3. User inputs racers' names.
4. Until user selects quit the game, user can play the game repeatedly.
5. User interface image is:

footracer_screenshot.png

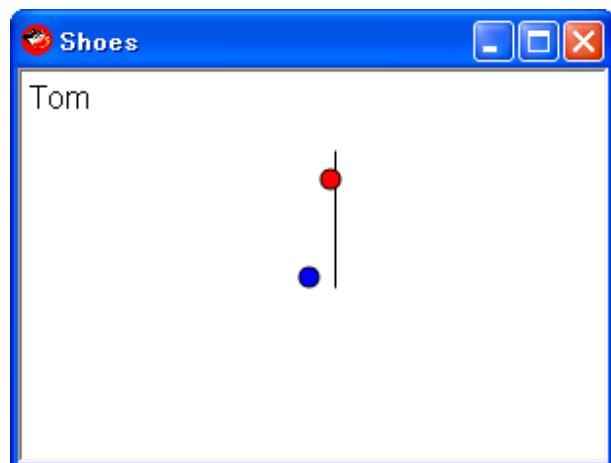


Have fun!

mini-footracer

There is the following two codes. They have the same behavior. Just write it as you like. :)

mini-footracer.png



The 1st one:

```

# mini-footracer-1st.rb
class Racer
  def initialize name
    @name, @avatar = name, nil
    @dist, @speed = 0, 15
  end

  attr_reader :name, :dist
  attr_accessor :avatar

  def run
    @dist += rand(@speed)
  end
end

class BabyRacer < Racer
  def initialize name
    super
    @dist, @speed = 100, 6
  end
end

tom = Racer.new 'Tom'
ash = BabyRacer.new 'Ash'

Shoes.app :width => 300, :height => 200 do
  line 160, 40, 160, 110
  tom.avatar = oval :fill => red, :left => 0, :top => 50, :radius => 5
  ash.avatar = oval :fill => blue, :left => 100, :top => 100, :radius =>
  a = animate do
    tom.avatar.move tom.run, 50
    ash.avatar.move ash.run, 100
    winner = tom.dist > ash.dist ? tom : ash
    (a.stop; para winner.name) if winner.dist > 150
  end
end

```

And the 2nd:

```

# mini-footracer-2nd.rb
Shoes.app :width => 300, :height => 200 do
  def run racer
    racer.left + rand(racer.style[:speed])
  end

  line 160, 40, 160, 110
  tom = oval :fill => red, :left => 0, :top => 50, :radius => 5, :speed
  ash = oval :fill => blue, :left => 100, :top => 100, :radius => 5, :sp

  a = animate do
    tom.move run(tom), 50
    ash.move run(ash), 100
    winner = tom.left > ash.left ? tom : ash
    (a.stop; para winner.style[:name]) if winner.left > 150
  end
end

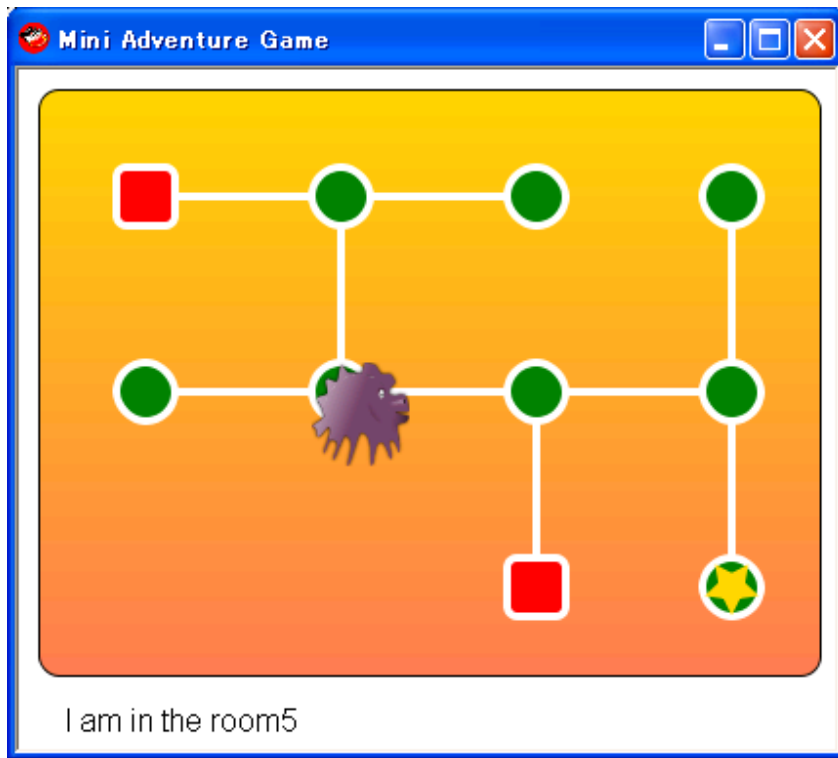
```

Assignment 3 Mini Adventure Game GUI Part

Create your own Mini Adventure Game GUI Part by doing the following 4 steps.

1. Create the adventure map
 - Place ten rooms on the map.
 - Entrance room and exit room have a different shape from the rest.
 - Treasure room has a star.
 - There are passages between rooms.
2. A treasure hunter appears on the map
 - At first, a treasure hunter appears in the Entrance room.
 - By pressing n/s/w/e on the keyboard, the hunter moves.
 - By pressing l, some messages will be shown.
3. hunter gets the treasure star
 - When the hunter enters the treasure room and t is pressed, the treasure star disappears.
4. Finish the adventure
 - When the hunter enters into the exit room with the star and l is pressed, the game ends.
 - After the hunter gets the star, they move jointly.

sample52.png



```
# sample52.rb
require 'sample52-render'

Shoes.app :width => 420, :height => 350, :title => 'Mini Adventure Game'
  extend Render

  show_map
  show_hunter

  keypress do |k|
    case k
    when 'n' then move_hunter 0, -100
    when 's' then move_hunter 0, 100
    when 'w' then move_hunter -100, 0
    when 'e' then move_hunter 100, 0
    else
    end and @msg.text = '' if can_go? k.to_s # Need to add .to_s for c

    case k
    when 'l'
      @msg.text = "I am in the #{room_name}"
      alert 'Congrats!' or exit if can_exit?
    when 't' then @msg.text = "Got a star!!" if got_star?
    else
    end
  end
end
end
```

See the below code, line 61: `@hunter.star 20, 30, 5, 10.0, 5.0, :fill => gold, :stroke => gold`
 This one line solution for the last demand is created by James Silberbauer.

```

# sample52-render.rb
module Render
  ROOMS =<<-EOS
entrance:e
room1:swe
room2:w
room3:s
room4:e
room5:nwe
room6:swe
room7:nsu
exit:n
room9:n
EOS

  def show_map
    @pos = [50, 50], [150, 50], [250, 50], [350, 50],
           [50, 150], [150, 150], [250, 150], [350, 150],
           [250, 250], [350, 250]

    fill gold.to_s..coral.to_s
    rect :width => 400, :height => 300, :left => 10, :top => 10, :curve
    stroke white
    strokewidth 4
    lines = [[0, 1], [1, 2], [1, 5], [4, 5], [5, 6], [6, 7], [7, 3], [6,
    lines.each{|a, b| line @pos[a][0] + 15, @pos[a][1] + 15, @pos[b][0]

    @rooms = @pos.collect{|x, y| rect x, y, 30, 30, :curve => 15, :fill
    [0, 8].each{|n| @rooms[n].style :fill => red, :curve => 5}

    ROOMS.each_with_index do |r, i|
      name, paths = r.chomp.split(':')
      @rooms[i].style :name => name, :paths => paths
    end

    @star = star 365, 265, 5, 10.0, 5.0, :fill => gold, :stroke => gold

    @msg = para '', :left => 20, :top => 320
  end

  def show_hunter
    @hunter = image '../images/loogink.png', :left => @pos[0][0], :top =
    @x, @y = 50, 50
  end

  def move_hunter x, y
    @hunter.move @x += x, @y += y
  end

  def can_go? k
    @rooms[@pos.index [@hunter.left, @hunter.top]].style[:paths].index k
  end

  def room_name
    @rooms[@pos.index [@hunter.left, @hunter.top]].style[:name]
  end

  def got_star?
    return false if @star.hidden
    if @hunter.left == @star.left - 15 and @hunter.top == @star.top - 15
      @star.hide
    end
  end
end

```

Have fun!

Assignment 4 Pong in Shoes

Create your own Pong in Shoes by doing the following 5 steps.

1. Open Shoes Window / Play Pong in Shoes
 - Window's width and height are both 400 pixel.
 - Can't resize.
 - Show your app name and revision number on the window's title bar.
 - Color the surface of the window with the horizontal gradation.
 - Play [Pong in Shoes](#) written by _why.
 - Hack (read) the code.
2. Show two paddles and a ball
 - Allocate computer paddle on the top (immobile yet).
 - Allocate player's (your) paddle on the bottom.
 - Your paddle synchronizes with the mouse movement.
 - A ball appears left-top side and moves smoothly to right-bottom side at 20 frames per second.
3. Lock-in the ball within the window
 - Bounce a ball on the edge of the window.
 - Computer's paddle synchronizes with the ball movement.
4. Hit the ball
 - Have your paddle hit the ball.
 - have computer's paddle hit the ball.
 - Change ball's speed and bounce angle when the ball is hit.
5. Have a match
 - When the ball goes over the goal lines, game finishes with victory message.

sample58.png




```

# sample58.rb
Shoes.app :width => 400, :height => 400, :resizable => false do
  vx, vy = 3, 4

  nostroke
  @ball = oval 0, 0, 20, :fill => forestgreen
  @comp = rect 0, 0, 75, 4, :curve => 2
  @you = rect 0, 396, 75, 4, :curve => 2

  @anim = animate 40 do
    nx, ny = @ball.left + vx.to_i, @ball.top + vy.to_i

    if @ball.top + 20 < 0 or @ball.top > 400
      para strong("GAME OVER", :size => 32), "\n",
        @ball.top < 0 ? "You win!" : "Computer wins", :top => 140, :align => :center,
        @ball.hide and @anim.stop
    end

    vx = -vx if nx + 20 > 400 or nx < 0

    if ny + 20 > 400 and nx + 20 > @you.left and nx < @you.left + 75
      vy = -vy * 1.2
      vx = (nx - @you.left - (75 / 2)) * 0.25
    end

    if ny < 0 and nx + 20 > @comp.left and nx < @comp.left + 75
      vy = -vy * 1.2
      vx = (nx - @comp.left - (75 / 2)) * 0.25
    end

    @ball.move nx, ny
    @you.left = mouse[1] - (75 / 2)
    @comp.left += 10 if @comp.left + 75 < @ball.left
    @comp.left -= 10 if @ball.left + 20 < @comp.left
  end
end

```

Have fun!

Note

[a pong challenge](#)

Assignment 5 Riddles in Shoes

Create your own Riddles in Shoes by doing the following 5 steps.

1. Open Shoes Window
 - Window's width and height are both 400 pixel.
 - Can't resize.
 - Show your app name and revision number on the window's title bar.
 - Color the surface of the window with a horizontal gradation.
2. Lay out titles and elements
 - Show title 'Riddles in Shoes'.

- Show line under the title.
 - Show subtitles 'Question', 'Answer' and 'Score'.
 - Align 10 stars under the line.
3. Show a riddle and create an input answer area
 - Click a button to show a riddle at random.
 - Use these riddles. (*1)
 4. Move down the star
 - Click a button to smoothly move down a star for the question.
 - If the answer is correct, align the star at the left-bottom.
 - If the answer is incorrect, align the star at the right-bottom.
 5. Play Riddles in Shoes
 - Finish the game after answering all riddles.
 - Decorate the surface to your taste.

sample67.png



```

# sample67.rb
Riddles =<<-EOS
what letter is a drink? --> t
what has nothing but a head and a tail? --> coin
what is it that by losing an eye has nothing left but a nose? --> noise
what bird can lift the heaviest weight? --> crane
what is broken when you name it? --> silence
what is a foreign ant? --> important
what lives on its own substance and dies when it devours itself? --> can
Yesterday is always before today. But there is a place where yesterday a
How many cookies can you eat on an empty stomach? --> one
what clothing does a house wear? --> address
EOS
Nums = (0..9).sort_by{rand}

Shoes.app :width => 400, :height => 410, :title => 'Riddles r0.5', :resi
  def set_riddle
    @num = Nums.pop
    alert('*waves*') or exit unless @num
    @q.text, @a = Riddles.to_a[@num].split(' --> ')
    @you.text = nil if @you
  end

  def set_score s
    x = @a.chomp == @you.text ? @i += 25 : @j += 25
    s.move x, 390
  end

  background orange..gold
  title 'Riddles in Shoes', :align => 'center'
  line 10, 55, 390, 55, :strokewidth => 5
  stars = []

  subtitle 'Question', :left => 10, :top => 90
  @q = tagline '', :left => 20, :top => 130, :width => 360, :stroke => w
  set_riddle

  subtitle 'Answer', :left => 10, :top => 260
  @you = edit_line :left => 20, :top => 310, :width => 200
  button 'OK', :left => 225, :top => 310 do
    s = stars[@num]
    a = animate do |i|
      s.move s.left, s.top + i
      (a.remove; set_score(s); set_riddle) if s.top > 320
    end
  end

  subtitle 'Score', :left => 10, :top => 340
  para 'good job', :left => 10, :top => 380, :width => 100, :stroke => w
  para 'how unlucky', :left => 210, :top => 380, :width => 100, :stroke
  @i, @j = 0, 200

  10.times{|i| stars << star(100+ 30*i, 80, 12, 10, 7)}
end

```

Have fun!

*1: riddles

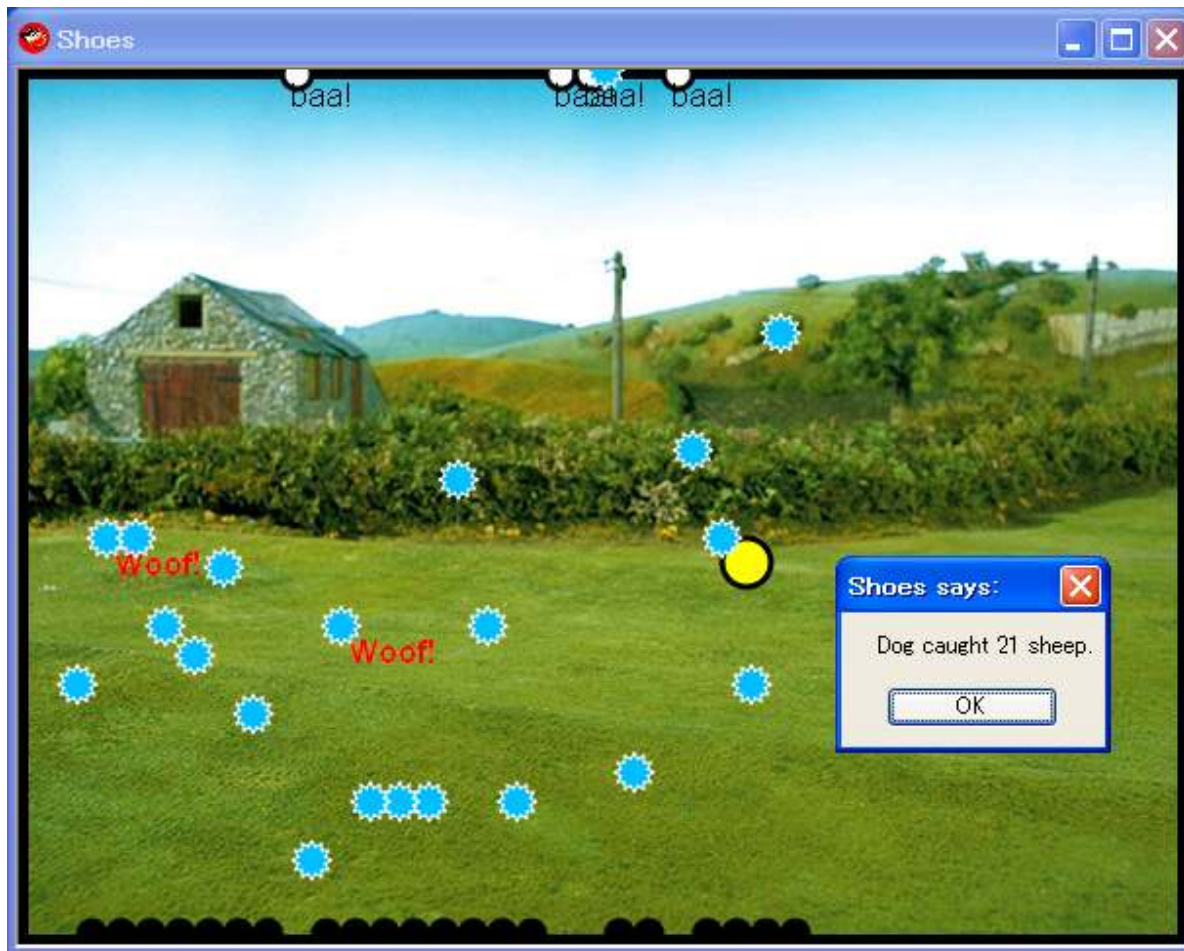
What letter is a drink? --> t
 What has nothing but a head and a tail? --> coin
 What is it that by losing an eye has nothing left but a nose? --> noise
 What bird can lift the heaviest weight? --> crane
 What is broken when you name it? --> silence
 What is a foreign ant? --> important
 What lives on its own substance and dies when it devours itself? --> candle
 Yesterday is always before today. But there is a place where yesterday always follows today.
 Where? --> dictionary
 How many cookies can you eat on an empty stomach? --> one
 What clothing does a house wear? --> address

Assignment 6 Dog Hunts Sheep Game

Create your own Dog Hunts Sheep Game by doing the following five steps.

1. A sheep running in the grass farm.
 - Use [this pic](#) as the background.
 - A sheep runs from the bottom of the window to the top.
 - The sheep stops at the top.
2. There are twenty-five sheep and one dog
 - Twenty-five sheep run at various speed or stop a while.
 - Dog moves with the arrow keys.
3. Dog woofs sheep
 - Dog woofs at sheep when and where he catches sheep.
 - 'Woofs!' will vanish after a while.
4. Dog catches sheep and sends them where they belong!
 - A deepskyblue star appears when and where the dog catches a sheep.
 - The sheep return where they belong.
5. Finish the game
 - Sheep say 'baa!' at the top.
 - The game finishes when all the sheep reach the top of the window or return the bottom.

sample68.png



```

# sample68.rb
w, h = 600, 450

class Sheep < Widget
  def initialize x
    @s = oval x, h - 15, 15, :fill => white, :strokewidth => 3
    @a = animate(20) do
      @s.move @s.left, @s.top - [0, 0, 15, 30, 45, 60][rand(6)]
      (@s.move(@s.left, -5); @a.stop; para 'baa!', :left => @s.left, :to
    end
  end

  def pos
    return @s.left, @s.top
  end

  def woofed
    @s.hide
    @a.remove
  end

  def hidden?
    @s.hidden
  end
end

Shoes.app :width => w, :height => h do
  def dog_catch_sheep s
    return if s.hidden?
    x, y = s.pos
    s.woofed
    msg = para 'woof!', :left => x, :top => y, :stroke => red, :weight =
    timer(1){msg.remove}
    star x, y, 12, 10, 7, :fill => deepskyblue, :stroke => white
    oval x, h - 15, 15
    @caught += 1
  end

  @caught = 0
  background '../images/pasture.jpg'
  border black, :strokewidth => 5
  sheeps = []
  25.times{|i| sheeps << sheep(30 + 15 * i)}

  dog = oval 300, 210, 25, :fill => yellow, :strokewidth => 3
  keypress do |key|
    x, y = 0, 0
    case key
      when :up then y = -15
      when :down then y = 15
      when :left then x = -15
      when :right then x = 15
      else
    end
    dog.move dog.left + x, dog.top + y
    sheeps.each{|s| dog_catch_sheep(s) if [dog.left, dog.top] == s.pos}
    reached = 0
    sheeps.each{|s| reached += 1 if s.pos[1] == -5}
    alert("Dog caught #{@caught} sheep.") or exit if @caught + reached =
  end
end

```

Have fun!

NOTE

Original article was created by [Karel Minarik](#). Thank you for giving us consent to use [the material](#).

Relevant web sites (Links)

Three manuals: Nobody Knows Shoes (NKS) and Built-in Manual and Online Reference Manual.

<http://shoooes.net/manuals/>

The Shoes Help Desk: The spot for beginners and advanced Shoesers alike.

<http://help.shoooes.net/>

The Shoebox

<http://the-shoebox.org/>

Rubyinside.com the latest article

Shoes - Rubys Cross Platform GUI App Toolkit - Grows Up

<http://www.rubyinside.com/whys-shoes-grows-up-1014.html>

RecentBuilds

<http://github.com/why/shoes/wikis/recentbuilds>

Shoes_(GUI_toolkit) in Wikipedia

[http://en.wikipedia.org/wiki/Shoes_\(GUI_toolkit\)](http://en.wikipedia.org/wiki/Shoes_(GUI_toolkit))

shoes_demonstration_apps

http://github.com/karmi/shoes_demonstration_apps/tree/master

Attempted refactoring a bit to learn more. ;-)

```

# sample62.rb
# See the following original code.
# 05_interactivity_with_objects.rb
# http://github.com/karmi/shoes_demonstration_apps/tree/master

class Letter < widget
  def initialize img
    @flag = false
    img.click{@flag = true; @img = img}
    img.release{@flag = false}
    motion[|left, top| @img.move(left-50, top-100) if @flag}
  end
end

Shoes.app :width => 800, :height => 600 do
  sound = video 'assets/drumfill.aif', :width => 0, :height => 0
  para 'Input Your Name: '
  spell = edit_line
  button 'GO' do
    sound.play
    @canvas.clear do
      spell.text.downcase.split('').each_with_index do |l, i|
        letter image("letters/#{l}.jpg", :left => 100*(i%8), :top => 15)
      end
    end
  end
end

@canvas = flow do
  a = 'a'
  26.times do |l|
    letter image("letters/#{a}.jpg").move(rand(width), rand(height))
    a.next!
  end
end
end

```

sample62.png



Appendix

Advanced articles

Threaded XMLHttpRequest In Shoes

<http://hackety.org/2008/08/15/threadedDownloadsInShoes.html>

Stamping EXEs And DMGs

<http://hackety.org/2008/06/19/stampingExesAndDmgs.html>

Martin DeMello's Gooey Challenge

<http://hackety.org/2008/06/12/martinDemellosGooeyChallenge.html>

The Image Block At The Bottom Of Shoes

<http://hackety.org/2008/05/22/theImageBlockAtTheBottomOfShoes.html>

Shoes mailing list in English

To join the mailing list:

Send a message to shoes AT code.whytheluckystiff.net

Cc: why AT whytheluckystiff.net

The archives are available at:

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/>

or

<http://news.gmane.org/gmane.comp.lib.shoes>

Shoes mailing list in Spanish

<http://groups.google.com/group/zapatos>

Shoes IRC channel

#shoes on irc.freenode.net

the Shoes adventurer's list

<http://code.whytheluckystiff.net/list/shoes/>

Acknowledgment

Authors:

Satoshi Asakaw (aka ashbb): main author

Michael Kohl (aka citizen428): corrections, additions, clarifications

Victor Goff (aka kotp): setup semi-automated prawn install.

Contributors:

- Peter corrected the shoes_course_text file.
- Michele corrected the ReadMeFirst file.
- Jerry corrected the whole Shoes Tutorial Note markdown files and created a handy tool, mkpdf.rb.
- Krzysztof, George, Sergio and Mareike gave some good ideas. They were very useful to create sample codes.
- George made good style sheets for html files and edited the tool, mkhtml.rb. He gave fancy gallery apps.
- Takaaki, Jose, Vic showed good tips.

Fancy Gallery

Gallery No.1

Listen - Ruby's top teacher, Satish Talim.

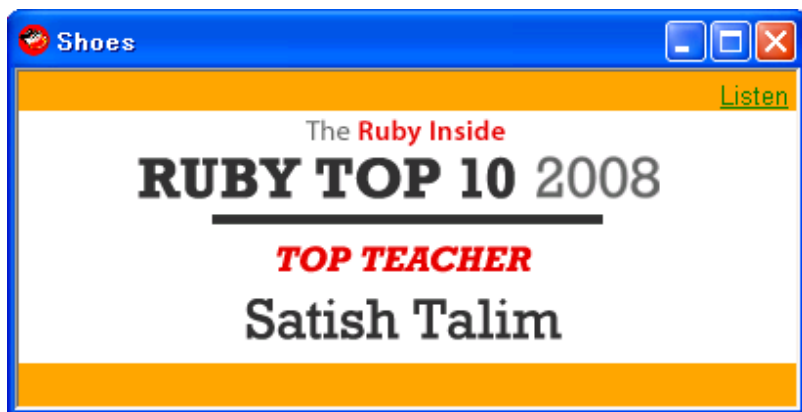
Original code was created by George and his grandson team. Cool!

```
# gallery1.rb
Shoes.setup do
  gem 'win32-sapi'
end

require 'win32/sapi5'

Shoes.app :width => 400, :height => 173 do
  background orange
  image '../images/rubytop10-teacher.gif', :top => 20
  inscription ins('Listen'), :align => 'right', :stroke => green
  words = "Ruby's top teacher, Satish Talim"
  click{win32::SpVoice.new.speak words}
end
```

gallery1.png



[Ruby's Top Teacher in 2008 - Satish Talim](#)

Gallery No.2

Simple custom edit box with background image.

Inspired the Eric Proctor's post in POIRPWSC101-2I

```
# gallery2.rb
Shoes.app :width => 200, :height => 200 do
  background mintcream, :width => 1.0, :height => 1.0
  @s = stack :margin => 5, :width => 1.0, :height => 1.0 do
    background '../images/shell.png', :curve => 5
    @line = para '', :stroke => white, :weight => 'bold'
    @line.cursor = -1
  end

  keypress do |k|
    case k
    when String, "\n"
      @line.text += k
    when :backspace
      @line.text = @line.text[0..-2]
    else
    end
  end
end
```

gallery2.png



Gallery No.3

Live code! Rewrite the code whatever you want! Change colors at once when you write correct code.

Inspired the George Thompson's post in POIRPWSC101-2I

```
# gallery3.rb
Shoes.app :title => "Live Code", :width => 500, :height => 240, :resizab
  background purple..white

txt =<<-EOS
  def get_random_color
    send COLORS.keys.map{|sym|sym.to_s}[rand(COLORS.keys.size)]
  end
  get_random_color
EOS

title 'Random Colors', :left => 10, :stroke => white
para "Rewrite the code whatever you want!\nChange colors at once\nwhen
  :left => 10 , :top => 60, :width => 540

code = edit_box txt , :left => 10 , :top => 140 , :width => 360 , :hei
  every(1){oval width - 130, 30, 100 , :stroke => eval(code.text), :stro
end
```

gallery3.png



Gallery No.4

Webcomic on Shoes! A simple app but quite convenient. Cool!

Original code was created by [Michael Kohl](#).

His code uses `shoes.setup` for requiring Mechanize and uses `image` with no arguments. The code works well on Mac. But unfortunately on Windows, Shoes 2 still has some problems. xx-P So, I try to come up with a solution to get around the problems. Here we go. :)

```

# gallery4.rb
Shoes.app :title => "ManWithHat - An xkcd viewer" do

  def get_strip_details page
    system "ruby gallery4-1.rb #{page}"
    @strip.path, @funny.text, @title.text = IO.readlines('xkcd_tmp.viewe
  end

  flow :margin => 10 do
    button 'Latest strip' do
      page = 'http://xkcd.org/'
      get_strip_details page
    end

    button 'Random strip' do
      page = 'http://dynamic.xkcd.com/comic/random/'
      get_strip_details page
    end

    stack do
      @title = caption
      @funny = para
      @strip = image '../images/space.png'
      @disclaimer = para em 'All cartoons and texts used in '\
                           'this application were made by xkcd.com.'
    end
  end
end

```

The following code is not a Shoes app, just a Ruby script to use Mechanize. ;-)

```

# gallery4-1.rb
require 'mechanize'

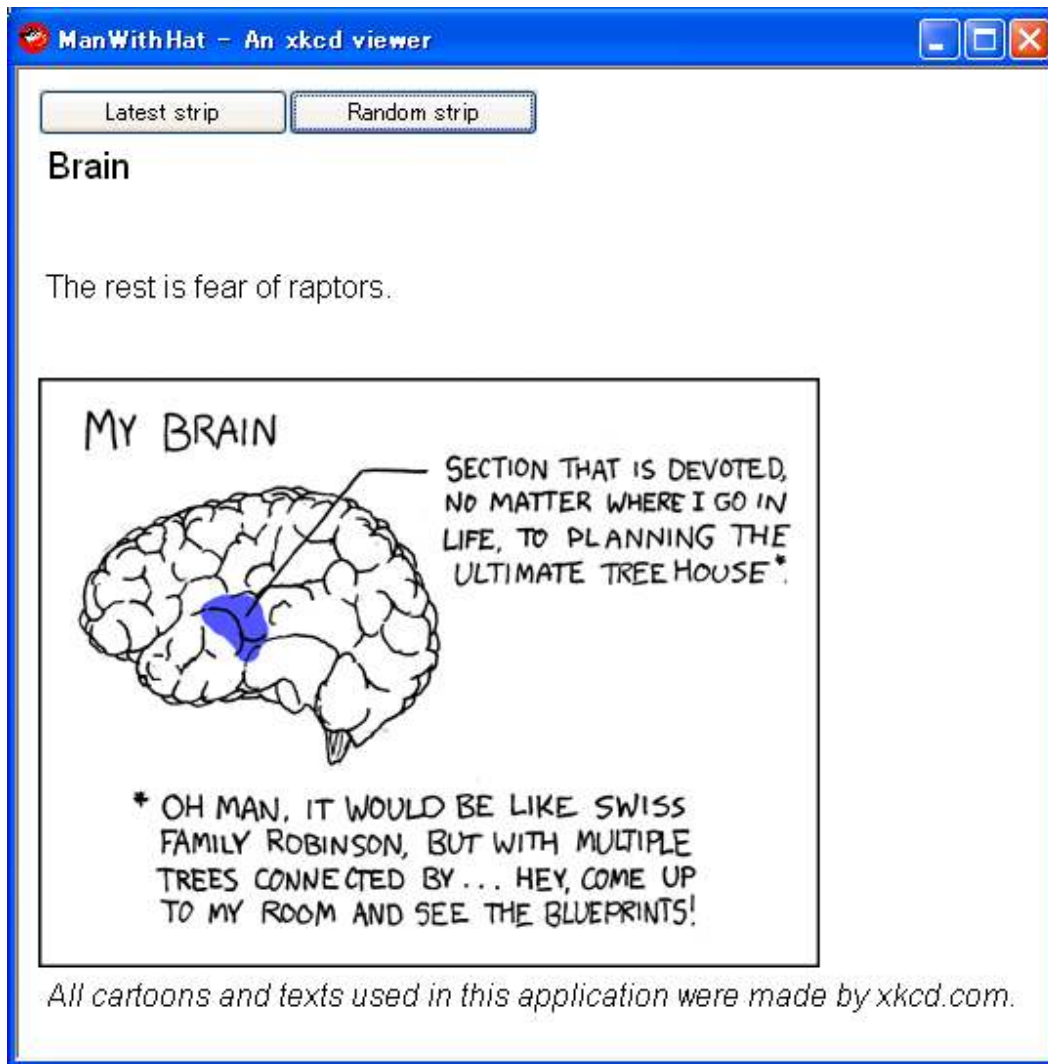
agent = WWW::Mechanize.new
page = ARGV[0]

page_content = agent.get(page)
image_regex = / : for BlueC

image_regex.match page_content.search("//img[contains(@src,'comics')]").
open('xkcd_tmp.viewer', 'w'){|f| f.puts $1, $2, $3}

```

gallery4.png



Gallery No.5

Do you know [Para-Para Manga](#)? That is called a flip book in English (from Wikipedia).

This is a very simple code. Just for fun. ;-)

The original gif animation is on this site:

[Ultrahigh speed potato chopping](#) - Japanese site.

```

# gallery5.rb
Shoes.app :width => 175, :height => 160 do
  background tan
  st = {:left => 10, :top => 10}

  @images = []
  1.upto 60 do |i|
    n = '00' + i.to_s
    n = n[-3..-1]
    @images << image("potato_chopping/1258_s#{n}.gif").hide
  end

  @images.each{|img| img.style st}

  def potacho
    @images[59].hide
    a = animate 12 do |i|
      @images[i].show
      @images[i-1].hide if i > 0
      a.stop if i > 58
    end
  end

  button 'start', :left => 10, :bottom => 0 do
    potacho
  end
end

```

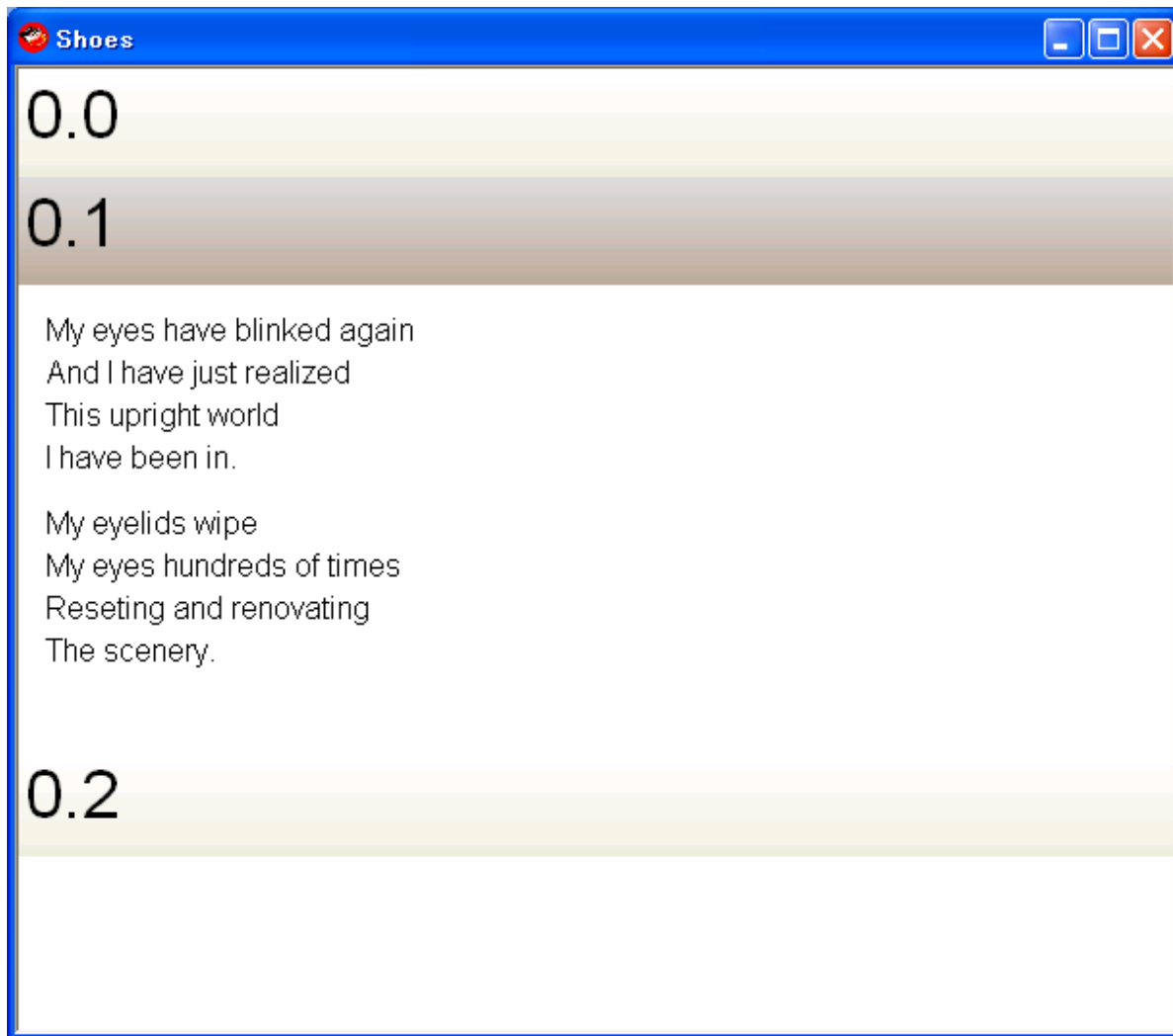
gallery5.png



Built-in Samples

simple-accordion

simple-accordion.png




```

# simple-accordion.rb
module Accordion
  def open_page stack
    active = app.slot.contents.map { |x| x.contents[1] }.
      detect { |x| x.height > 0 }
    return if active == stack
    a = animate 60 do
      stack.height += 20
      active.height = 240 - stack.height if active
      a.stop if stack.height == 240
    end
  end
  def page title, text
    @pages ||= []
    @pages <<
      stack do
        page_text = nil
        stack :width => "100%" do
          background "#fff".."#eed"
          hi = background "#ddd".."#ba9", :hidden => true
          para link(title) {}, :size => 26
          hover { hi.show }
          leave { hi.hide }
          click { open_page page_text }
        end
        page_text =
          stack :width => "100%", :height => (@pages.empty? ? 240 : 0) d
          stack :margin => 10 do
            text.split(/\n{2,}/).each do |pg|
              para pg
            end
          end
        end
      end
    end
  end
end

Shoes.app do
  extend Accordion
  style(Link, :stroke => black, :underline => nil, :weight => "strong")
  style(LinkHover, :stroke => black, :fill => nil, :underline => nil)

  page "0.0", <<- 'END'
  There is a thought
  I have just had
  which I dont care to pass to
  Anyone at all at this time.

  I have even forgotten it now,
  But kept only the pleasures
  Of my property
  And of my controlled mental slippage.
  END
  page "0.1", <<- 'END'
  My eyes have blinked again
  And I have just realized
  This upright world
  I have been in.

  My eyelids wipe

```

Study Note

#1:

- The `app.slot` is the Shoes window itself. It's a flow.
- The contents is the Shoes method lists all elements in a slot. Refer to: <http://help.shoooes.net/Traversing.html>
- `x.contents[1]` is the stack which is defined at #9. `x.contents[0]` is the stack which is defined at #6.

#2:

See `ri Enumerable#detect`.

#3:

This stack is a local variable (an argument of `open_page` method)

#4:

This active is a page clicked.

#5:

Same as the following.

```
s = stack do
  # bla bla bla
end
@pages << s
```

#8:

In this case, there is no need to use a link. It's enough just like this:

```
para title, :size => 26
```

Because the perception about mouse hover/leave is doing with background element (#7).

#9:

If you don't need to open the first page as a default, just write like this:

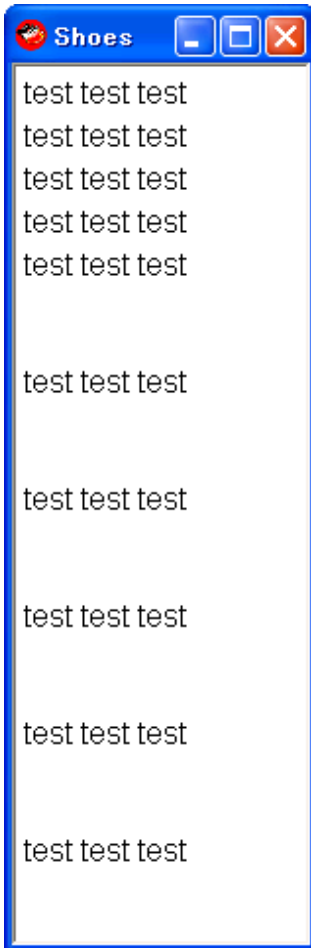
```
stack :width => "100%", :height => 0 do
```

#11:

Why split by two new lines? Do the following snippet.

```
Shoes.app :width => 150, :height => 450 do
  flow do
    5.times{para "test test test\n"}
  end
  stack do
    5.times{para "test test test\n"}
  end
end
```

simple-accordion-study-note-snippet-1.png



There is a difference between stack and flow. Because...

<http://www.mail-archive.com/shoes@code.whytheluckystiff.net/msg02869.html>

In this case, we can write like this instead of #10 and #11.

```
flow :margin => 10 do
  text.each do |pg|
```

#12:

The extend method is used to add two methods, open_page and page, as instance methods into Shoes.app object. See `ri object#extend`.

#13 and #14:

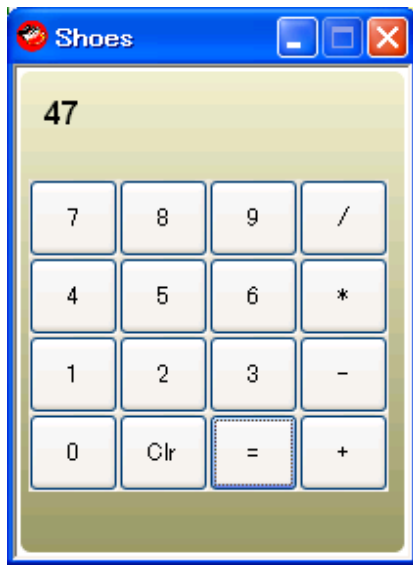
Change the default style for the link method. But there is no need to use a link in this case. See the above explanation about #8.

#15:

Using [here document](#). Check the usage of '(single quote)', "(double quote)" and -(hyphen).

simple-calc

simple-calc.png



```

# simple-calc.rb
class Calc                                     #1
  def initialize                               #2
    @number = 0
    @previous = nil
    @op = nil
  end

  def to_s                                     #3
    @number.to_s
  end

  (0..9).each do |n|                           #4
    define_method "press_#{n}" do
      @number = @number.to_i * 10 + n
    end
  end

  def press_clear
    @number = 0
  end

  {'add' => '+', 'sub' => '-', 'times' => '*', 'div' => '/'}.each do |meth|
    define_method "press_#{meth}" do
      if @op                                     #6
        press_equals
      end
      @op = meth
      @previous, @number = @number, nil          #7
    end
  end

  def press_equals                             #8
    @number = @previous.send(@op, @number.to_i)
    @op = nil
  end

end

number_field = nil
number = Calc.new                             #9
Shoes.app :height => 250, :width => 200, :resizable => false do
  background "#EEC".."#996", :curve => 5, :margin => 2

  stack :margin => 2 do

    stack :margin => 8 do                       #10
      number_field = para strong(number)
    end

    flow :width => 218, :margin => 4 do
      %w(7 8 9 / 4 5 6 * 1 2 3 - 0 Clr = +).each do |btn|
        button btn, :width => 46, :height => 46 do #11
          method = case btn
            when /[0-9]/: "press_#{btn}"
            when 'Clr': 'press_clear'
            when '=': 'press_equals'
            when '+': 'press_add'
            when '-': 'press_sub'
            when '*': 'press_times'
          end
        end
      end
    end
  end
end

```

Study Note

#1:

Define a class Calc. This class has the following 18 methods.

- initialize
- to_s
- press_0, press_1, , press_9
- press_clear
- press_add, press_sub, press_times, press_div
- press_equal

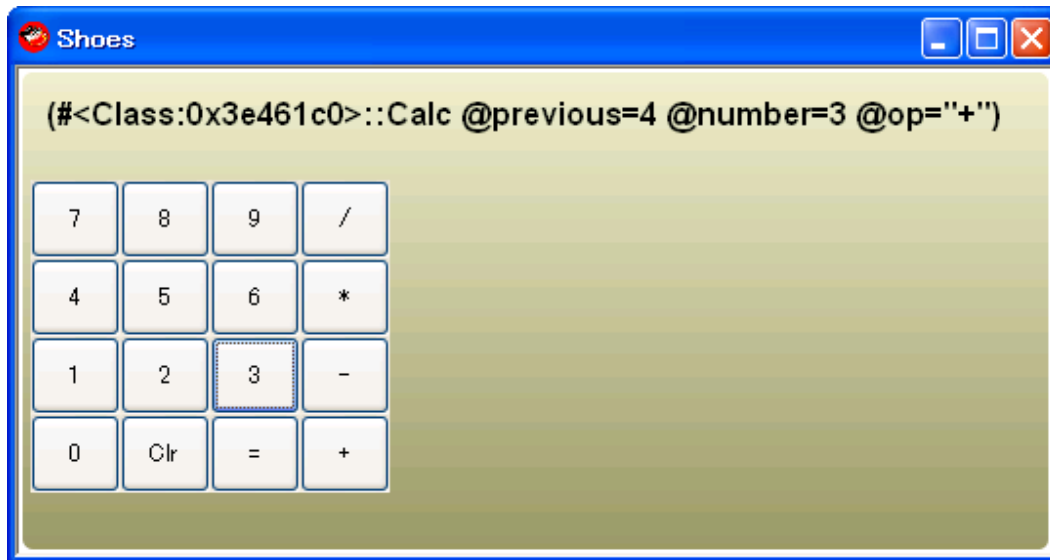
#2:

Define a method initialize. This method is used once in the above code to create just one object at #9.

#3:

Define a method to_s. In Shoes, para uses to_s implicitly at #10. If you comment out to_s definition lines, you will see the following output.

simple-calc-1.png



#4:

Define methods press_0, press_1, , press_9 with the method define_method. See ri Module#define_method. They are the same as the following.

```
def press_0
  @number = @number.to_i * 10 + 0
end
def press_1
  @number = @number.to_i * 10 + 1
end
:
: (and so on)
```

The .to_i is necessary for the case that @number is nil.

#5:

Define methods press_add, press_sub, press_times, press_div.

#6:

Execute the previous calculation (one of +, -, *, /).

#7:

To clear the `number_field` at #13 (to show nothing), assign `nil` to `@number` instead of 0.

#8:

Define a method `press_equals`. See the following small IRB snippet.

```
C:\>irb --simple-prompt
>> 23.send '+', 5
=> 28
```

#12:

Send the value (character string) that was assigned to the local variable `method` to the object `number` that was created at #9. See the following small IRB snippet.

```
C:\>irb --simple-prompt
>> class Calc
>>   def press_add
>>     puts 'DEBUG: hi.'
>>   end
>> end
=> nil
>> Calc.new.send 'press_add'
DEBUG: hi.
=> nil
>>
```

#13:

This line is defined within the button definition block (#11), hence every time when any button is clicked, `number_field` area will be refreshed.

Alternative Simple Calc

This is an alternative code. There is no interesting tips but it's simple, I guess. :)

```

#sample61.rb
Shoes.app :height => 250, :width => 200, :resizable => false do
  def do_calc
    @number = @previous.send(@op, @number) if @op
    @op = nil
  end

  @previous, @number, @op = 0, 0, nil

  background "#EEC".."#996", :curve => 5, :margin => 2

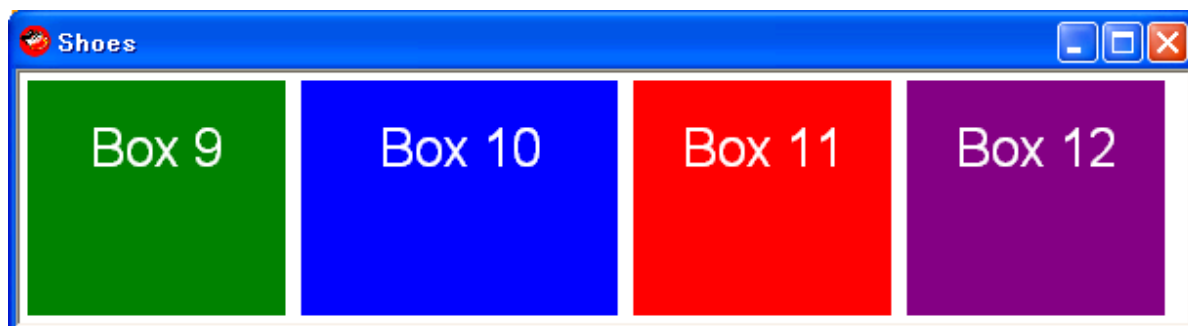
  stack :margin => 2 do
    stack :margin => 8 do
      @number_field = para strong(@number)
    end

    flow :width => 218, :margin => 4 do
      %w(7 8 9 / 4 5 6 * 1 2 3 - 0 C\r = +).each do |btn|
        button btn, :width => 46, :height => 46 do
          case btn
          when /[0-9]/
            @number = @number.to_i * 10 + btn.to_i
          when 'C\r'
            @previous, @number, @op = 0, 0, nil
          when '='
            do_calc
          else
            do_calc
            @previous, @number = @number, nil
            @op = btn
          end
        end
        @number_field.replace strong @number
      end
    end
  end
end
end
end
end
end

```

simple-menu

simple-menu-r1.png



This snapshot shows the following.
 hovered on blue box --> the width is expanded to 170 pixel.
 clicked twice --> Box numbers are replaced from 1-4 to 9-12.


```

# simple-menu-r1.rb
class MenuPanel < Shoes::Widget
  @@boxes = []
  def initialize(color, args)
    eval "self.style " + args
    @@boxes << self
    background color
    para link("Box #{@boxes.length}", :stroke => white, :fill => nil).
      click { visit "/" },
      :margin => 18, :align => "center", :size => 20
    hover { expand }
  end
  def expand
    if self.width < 170
      a = animate 30 do
        @@boxes.each do |b|
          b.width -= 5 if b != self and b.width > 140
        end
        self.width += 5
      end
      a.stop if self.width >= 170
    end
  end
end
end
end

Shoes.app :width => 600, :height => 130 do
  style(Link, :underline => nil)
  style(LinkHover, :fill => nil, :underline => nil)
  menu_panel green, ":width => 170, :height => 120, :margin => 4"
  menu_panel blue, ":width => 140, :height => 120, :margin => 4"
  menu_panel red, ":width => 140, :height => 120, :margin => 4"
  menu_panel purple, ":width => 140, :height => 120, :margin => 4"
end

```

Study Note

Original built-in sample code is [here](#).

But it didn't work well as is, so added #3 and modified from #7 to #11.

#1:

There is no need adding `Shoes::` explicitly. The following is also okay.

```
class MenuPanel < widget
```

#2:

When the initialize method of Widget subclass has hash as the argument, it seems including more than the number of given arguments. See the following:

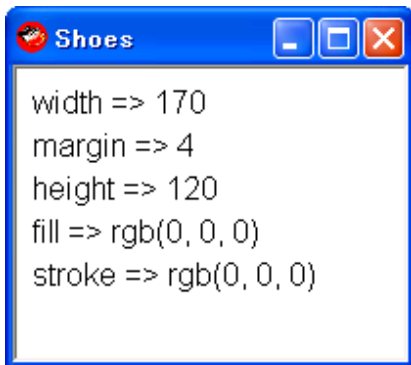
```

class MenuPanel < widget
  def initialize args
    data = ''
    args.each{|k, v| data << "#{k} => #{v}\n"}
    para data
  end
end

Shoes.app :width => 200, :height => 150 do
  menu_panel :width => 170, :height => 120, :margin => 4
end

```

simple-menu-study-note-snippet-1.png



#3:

I don't know why the following two snippets show the different outputs. It seems a bug, though...

```

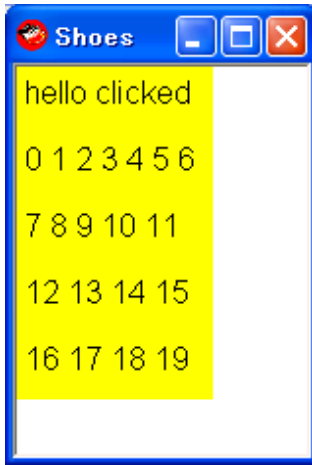
class Menu < widget
  def initialize #args
    self.style :width => 100, :height => 100
    background yellow
    para 'hello'
    click{ expand }
  end

  def expand
    para 'clicked'
    animate{|i| para(i) if i < 20}
  end
end

Shoes.app :width => 150, :height => 200 do
  #menu :width => 300, :height => 300
  menu
end

```

simple-menu-study-note-snippet-2.png



```

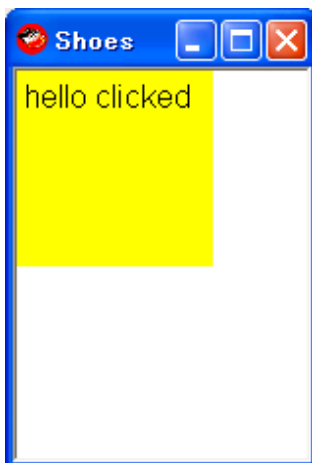
class Menu < widget
  def initialize args
    self.style :width => 100, :height => 100
    background yellow
    para 'hello'
    click{ expand }
  end

  def expand
    para 'clicked'
    animate{|i| para(i) if i < 20}
  end
end

Shoes.app :width => 150, :height => 200 do
  menu :width => 300, :height => 300
  #menu
end

```

simple-menu-study-note-snippet-3.png



#4 and #5:

If you replace `click { visit "/" }` to `click { }`, the Box number will not change. I guess `visit "/"` means the same as the refresh operation.

#6:

The method `expand` does the following.:

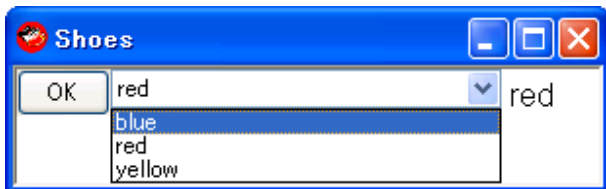
When mouse is hovering on a box which width is 140 pixel, the box expands till 170 pixel. At the same time, the box that has 170 pixel width rolls back to 140 pixel.

Trivia

list_box needs to set :height explicitly

```
# sample91.rb
Shoes.app :width => 300, :height => 60 do
  button('OK'){@msg.text = @e.text}
  @e = list_box :items => ['blue', 'red', 'yellow'], :height => 30
  @msg = para ''
end
```

sample91.png



Try to comment out :height => 30 and run.

The list_box doesn't show the items.

This strange behavior occurs only on Windows. On Mac OS X, it doesn't.

This OS X information was provided by George Thompson.

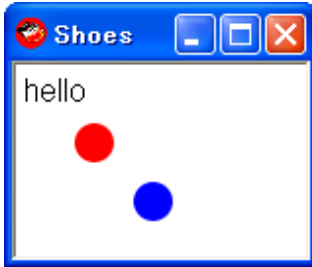
strange mouse event behavior

```
# sample92.rb
Shoes.app :width => 150, :height => 100 do
  @msg = para ''
  nostroke

  @img = image :width => 20, :height => 20, :left => 30, :top => 30 do
    oval :radius => 10, :fill => red
  end
  @img.hover{ @msg.replace 'hello' }
  @img.leave{ @msg.replace '' }

  @o = oval :left => 60, :top => 60, :radius => 10, :fill => blue
  @o.hover{ @msg.replace 'hi' }
  @o.leave{ @msg.replace '' }
end
```

sample92.png



The image (red) oval works the mouse hovering feature but the blue doesn't. This behavior is a bug. But it is fixed in the latest Shoes-0.r970 and later.

Shoes Fest

<http://shoes.yapok.org/>

Shoes was born July 31st, 2007.

Yes, July 31st is Shoes' birthday and it is now one year old.

Shoes wiki

A new Shoes wiki was launched on Sep 12th, 2008.

<http://github.com/why/shoes/wikis>

The old one was retired. Now linked to the Shoes Official Homepage.

<http://code.whytheluckystiff.net/shoes/>

<http://shooooes.net/>

Built-in sample apps

See the following directory (in Windows XP with Shoes-0.r1057)
There are many sample code. Let's hack!

C:\Program Files\Common Files\Shoes\0.r1057\samples

Building Shoes

If you have to build Shoes by yourself, this information might be useful.

<http://github.com/why/shoes/wikis/buildingshoes>

The Rules Of Shoes and UTF-8 Everywhere

Shoes scope can be a bit confusing...

Shoes supports UTF-8 everywhere. Can't wait to get the next build.

<http://newwwws.shooooes.net/2008/09/22/the-rules-of-shoes.html>

A very decent intro to shoes for beginners

<http://ruby.about.com/od/shoes/Shoes.htm>

Lovely creatures

Lovely creatures in this tutorial were created by Anita Kuno.

Each creature has his/her own name.

purple is loogink

green is Cy

brown is Yar

blue is kamome

white is shaha

```

# sample93.rb
Shoes.app :width => 400, :height => 75, :title => 'Lovely Creatures' do
  background "#D0A".."#F90", :angle => 90
  x = 0
  creatures = %w(loogink yar cy kamome shaha).collect{|c| image "../imag

  messages =<<-EOS
  Thx for reading. :)
  See you!
  Enjoy Ruby and Shoes!
  EOS
  messages = messages.to_a

  msg = subtitle '', :top => 30, :stroke => white
  animate(3) do
    creatures.each{|c| c.move c.left, rand(15)}
  end

  creatures.each do |c|
    c.hover{msg.text = strong messages[rand(messages.length)]}
    c.leave{msg.text = ''}
  end
end
end

```

sample93.png



Let's enjoy Ruby and Shoes with the Lovely Creatures!

FIN.