Shoes Tutorial Note

- For the Shoes App Rookie Creators -

Jan. 10th, 2009 by ashbb (Satoshi Asakawa)

Table of contents

Change log:

Jan 10th, 2009: Added new chapter Built-in Samples and the link to Shoes_(GUI_toolkit) in Wikipedia.
Jan 03rd, 2009: Added sample 60 and chapter 00535.
Dec 28th, 2008: Added sample 59 and chapter 00534.
Dec 25th, 2008: Edited Assignment 3 and added Assignment 4.
Dec 17th, 2008: Added gallery 3 and a hot topic.
Dec 13th, 2008: Added gallery 2.
Dec 12th, 2008: Added sample 55, 56, 57 and chapter Fancy Gallery.
Dec 08th, 2008: Added sample 54 and chapter 00533. Refactored mkmdown.rb
Dec 07th, 2008: Deleted patches and patch.rb
Dec 06th, 2008: Reviewed `shoes -p` with Shoes 2 (Raisins, 0.r1134)
Dec 06th, 2008: Replace image files for Shoes-0.r1123. Edited sample 42. Updated mkpdf.rb to the latest version. Added the link to RecentBuild page.
Dec 03rd, 2008: Added sample 53 and chapter 00532.
Dec 01st, 2008: Added '\' in front of '_' for githb spec change or bug.
Nov 29th, 2008: Modified file name '_' to '-' for github spec change or bug. Added assignment 3.
Nov 24th, 2008: Added a new tool, patch.rb, which replace image-file-path-on-github. This is a temporary patch. Because I'm not sure about a github spec change or a bug.
Nov 23th, 2008: Improved mkmdown.rb to add sample program names in the table of contents. Added a new tool, mkpdf.rb.
Nov 22th, 2008: Added a new chapter 00531. Added a link to the Shoes adventurer's list into chapter 00900.
Nov 20th, 2008: Added sample 49 into chapter 00508 and sample 50 into chapter 00513.
Nov 18th, 2008: Excuted all samples with Shoes-0.r1091 and updated some .png files and edited some .mdown files.

Nov 17th, 2008: Added a new tool, mkbightml.rb, which make one big html file included whole contents of Shoes Tutorial Note. Now, just a trial revision.
Nov 16th, 2008: Merge a 'browser' feature (side list of contents) provided by George.
Nov 15th, 2008: Added four tips into chapter 00300. Improved mkmdown.rb and mkhtml.rb to treat .jpg files and to treat page-links. Added more explanation into chapter 00504
Nov 14th, 2008: Fixed a bug in mkhtml.rb. Added a note into chapter 00528 and 00530 about BlueCloth's and Code Highlighter's bug.
Nov 13th, 2008: Added a new sample 48 into chapter 00530.
Nov 08th, 2008: Added Code Highlighter for html files.
Nov 07th, 2008: Added a new sample 47.
Nov 02st, 2008: Totally corrected English. Added acknowledgments, sample 46. Modified built-in constants
Nov 01st, 2008: Corrected typo of sample 31
Oct 30th, 2008: Added a hot topic.
Oct 29th, 2008: Added a new sample 45.
Oct 28th, 2008: Modified the page of 'Shoes.setup'
Oct 27th, 2008: Added a new sample 44.
Oct 26th, 2008: Modified sample 8, changed the path of image.
Oct 24th, 2008: Revised tools and modified .mdown files for the easy eBook maker.
Oct 23th, 2008: Added a new sample 43.

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 cross-platform tiny 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 NKS (Nobody Knows Shoes, The first public manual of Shoes. See chapter 8.)

Download Shoes

Download Shoes from this web site and pick the installer.

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

First step

There is a tutorial written by _why.

Now, copy and paste the whole 16 sample programs and run one by one. No need to understand the code meaning. Just run and look at the app window. This tutorial has screenshots, but be sure to run all 16 samples. Not later. Do it now, before going to the next, please. This is the most important step, I believe.

A Tip: Quick Launcher

Create .bashrc file and add this alias:

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

Now you can type in a terminal window:

```
shoes sample1.rb
```

Now you can quickly launch a shoes app without going through the open file dialog.

This tip is provided by George Thompson - Saturday, 15 November 2008, POIRPWSC101-1I

A Tip: Quick Launcher

The command to work with TextMate(http://samuraicoder.net/shoes.mov) # This link is not available now.

This tip is provided by Takaaki Kato - Saturday, 15 November 2008, POIRPWSC101-1I

A Tip: Quick Launcher

The windows link in a "Tool Bar".

shoes_launcher.jpg

This tip is provided by Victor Goff -- Saturday, 15 November 2008, POIRPWSC101-1I

A Tip: to Ubuntu 8.10 users

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 is provided by Jose Carlos Monteiro - Saturday, 15 November 2008, POIRPWSC101-1I

Birds-eye view (Survey basic features)

Concept

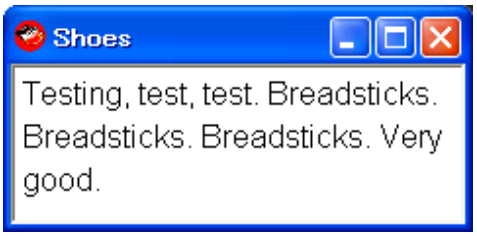
Shoes is a tiny graphics toolkit. It's simple and was born to be easy! So, Shoes doesn't have many elements (like tabbed controls, toolbars, horizontal scrollbars.) But they can be simulated with images. There are ten essential methods to know to get going with Shoes.

No. 1: para

banner : Charactor size 48 pixels
title : 34
subtitle : 26
tagline : 18
caption : 14
para (paragraph) : 12
inscription : 10

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

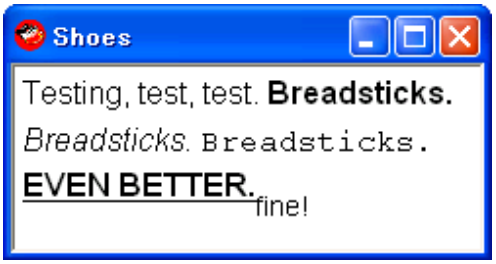
sample1.png



strong : bold
em (emphasized) : italics
code : monospace font
ins (inserted) : single underline
sub (subscript) : lower 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, read the following web page: http://github.com/why/shoes/wikis/stacksandflows

But use (run) the following sample code instead of the one on the above web page because the method Shoes#text is obsolete and you need to correct the path to image file.

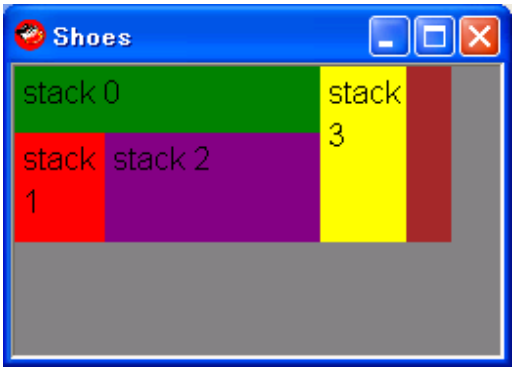
```
# sample3.rb
Shoes.app do
  stack do
    image "http://hackety.org/images/hackety-org-header.png"
  end
  stack :width => 200 do
    para "column 1"
  end
  stack :width => -200 do
    para "column 2"
  end
end
```

sample3.png

More complex sample code:

```
# 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

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

sample4.png



No. 4: button

button("Press Me")
which creates a new button and
button("Press Me"){ alert("clicked")}
the button fires the block when clicked and
button("Press Me", :left => 50, :top => 20)
will place the button at coordinates (50, 20).
That's it.

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

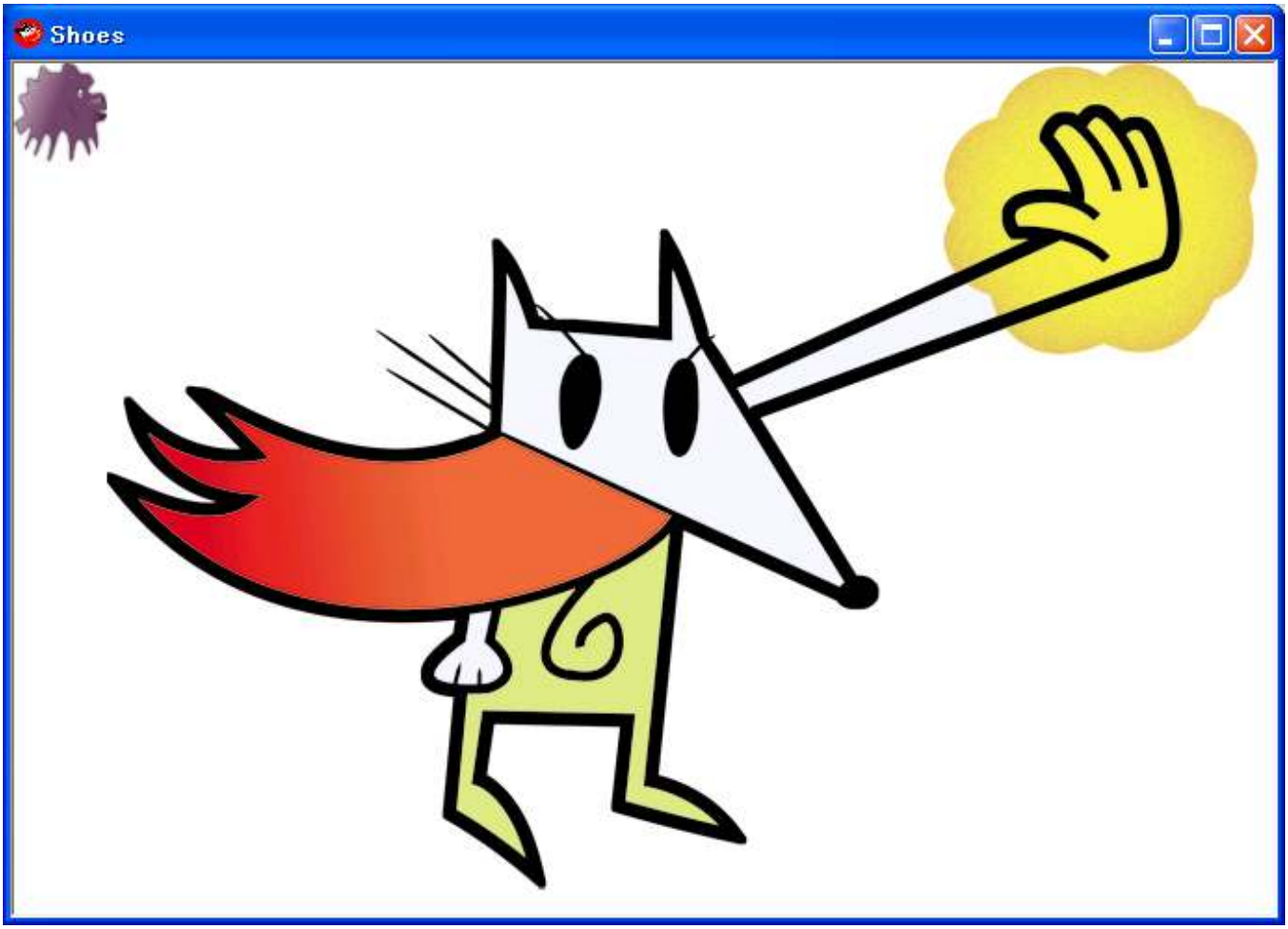
sample5.png

No. 5: image

An image is a picture in PNG, JPEG or GIF format. We can use the directory path or URL.

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

sample6.png



No. 6: edit_line

Edit boxes are wide, rectangular boxes for entering text. Edit lines are a slender, little box for entering one line of text.

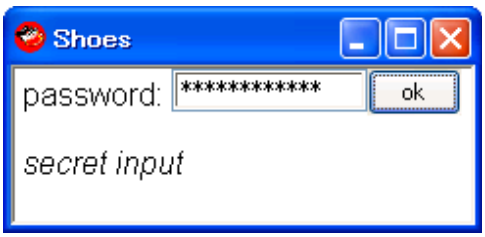
```
# sample7.rb
Shoes.app :width => 250, :height => 300 do
  stack do
    @msg = para 'Hello'
    @el = edit_line "We love Ruby."
    button('ok'){ @msg.text = @el.text}
    @eb = edit_box "We love Shoes."
    button('ok'){ @msg.text = @eb.text}
  end
end
```

sample7.png

We can use :secret in the edit_line area.

```
# 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

Hyperlinks. We have three ways to write the links.

```
# 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.
Background blue, :radius => 12
But it is obsolete. Now we can use :curve instead of :radius. And can also use :angle for gradient.

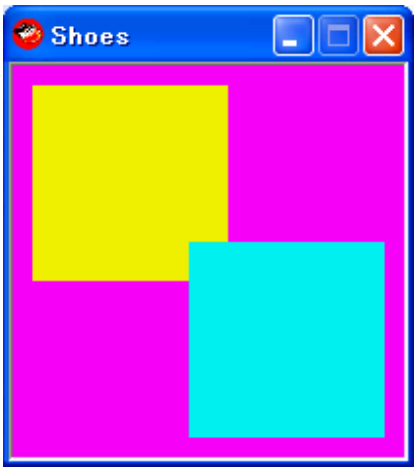
```
# sample10.rb
Shoes.app :width => 200, :height => 70 do
  background "#D0A"..darkorange.to_s, :angle => 45, :curve => 30
end
```

sample10.png

We can change the background colors.

```
#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



changeing colour of background

No. 9: Shoes.url

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. From the user viewpoint, it just behaves like a page on the web.

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

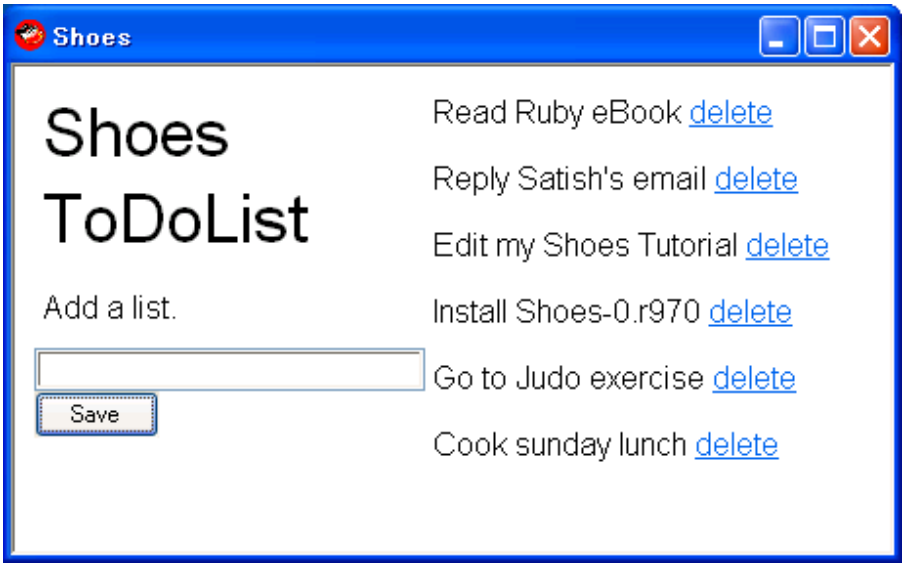
```
# sample12.rb
Shoes.app :title => 'RC', :width => 100, :height => 80 do
  def random_creatures
    background rgb rand(256), rand(256), rand(256)
    name = %w[loogink cy yar kamome shaha][rand 5]
    image '../images/' + name + '.png', :left => 30, :top => 10
  end

  random_creatures

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

sample12.png

The append and remove methods are also useful.

```
# 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 original Shoes apps

Open Shoes built-in manual and Shoes console window

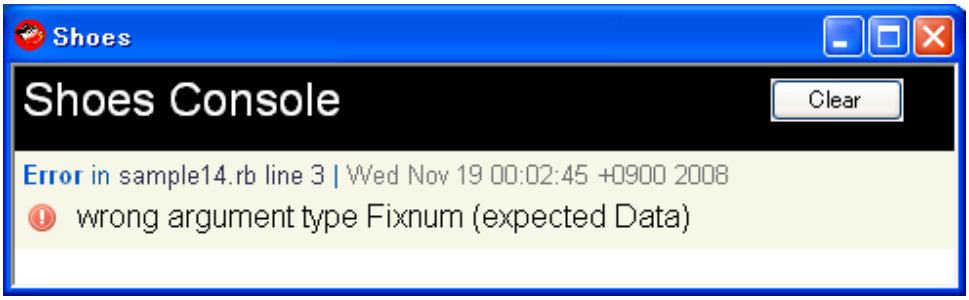
To open the Shoes built-in manual, Type the following on your pc console (terminal window).

shoes -m
or
shoes --manual

Or type Alt + ? on any Shoes app window.
Or select from the menu. See here.
http://shoooes.net/manuals/

To open the Shoes console window,
type Alt + / on any Shoes app window.

shoes_console.png

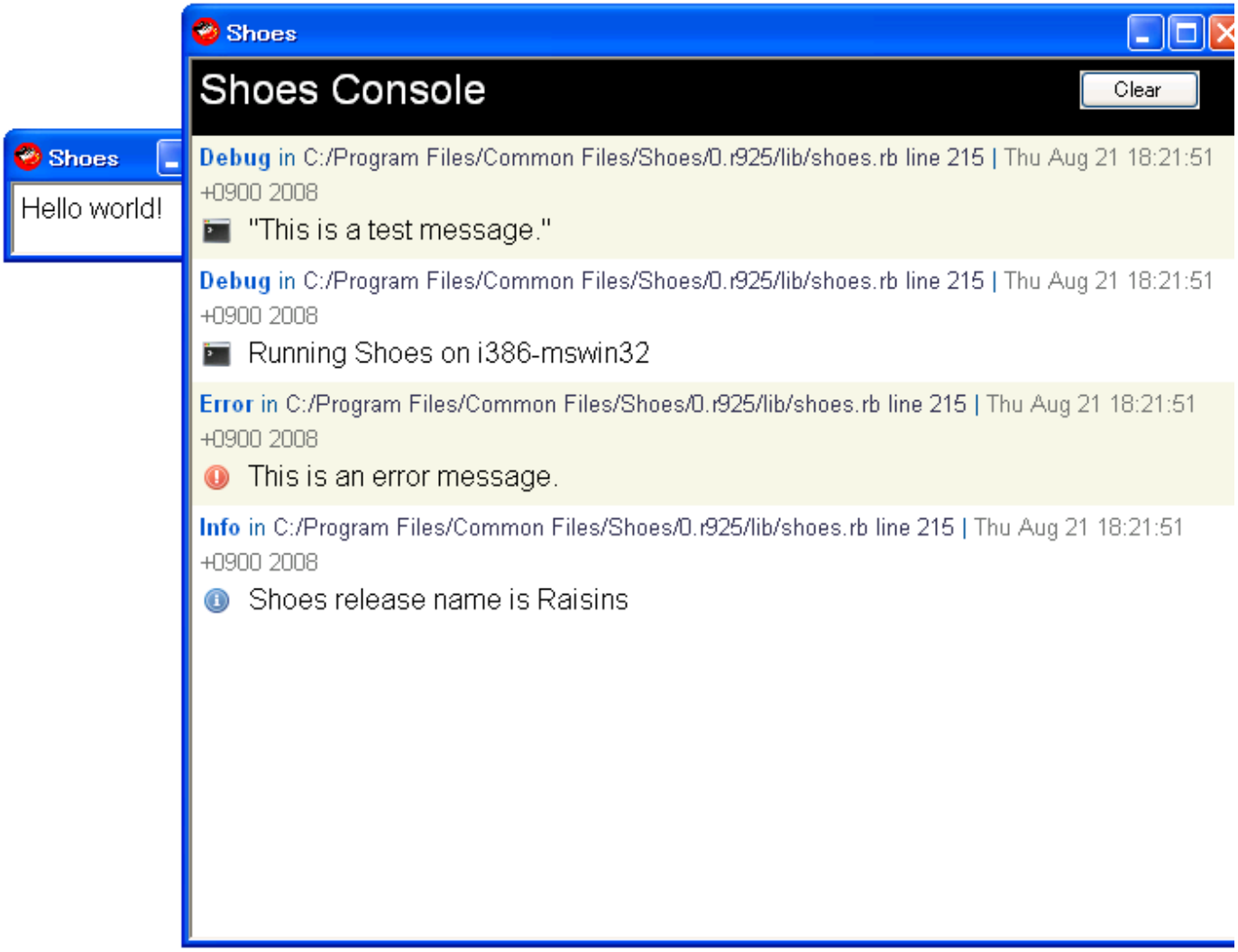


Output messages on the Shoes console window

We 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



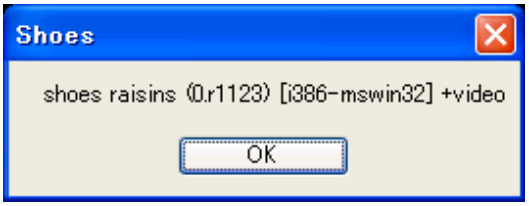
shoes --help

Type the following on your pc console (terminal window).

shoes -h
or
shoes --help

shoes_help.png

shoes_version.png

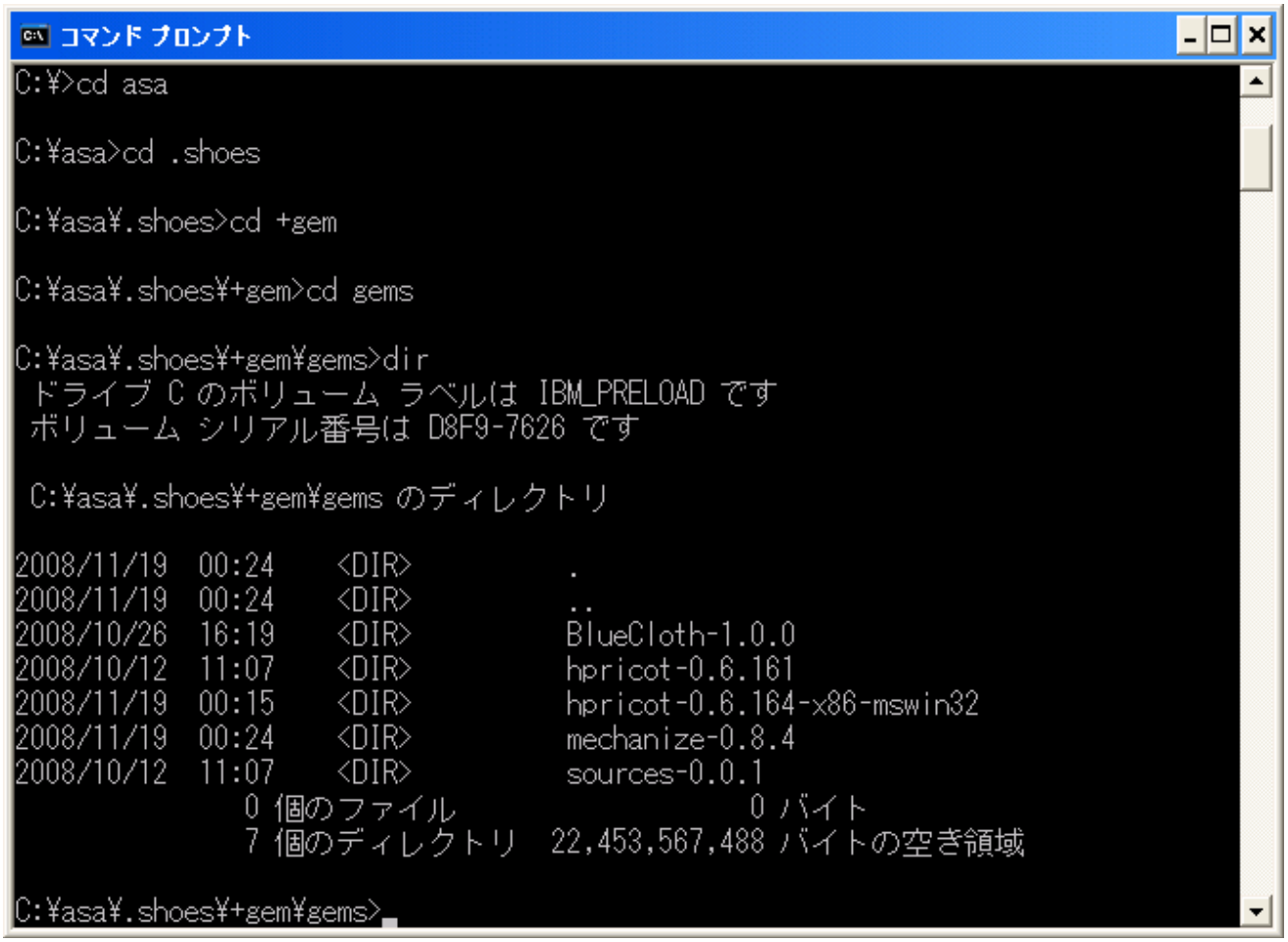


shoes_download_test.png

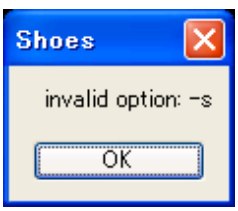
shoes_gem.png

shoes -g install hpricot
shoes -g install mechanize
With Shoes-0.r1091, it works well!

shoes_gem-1.png

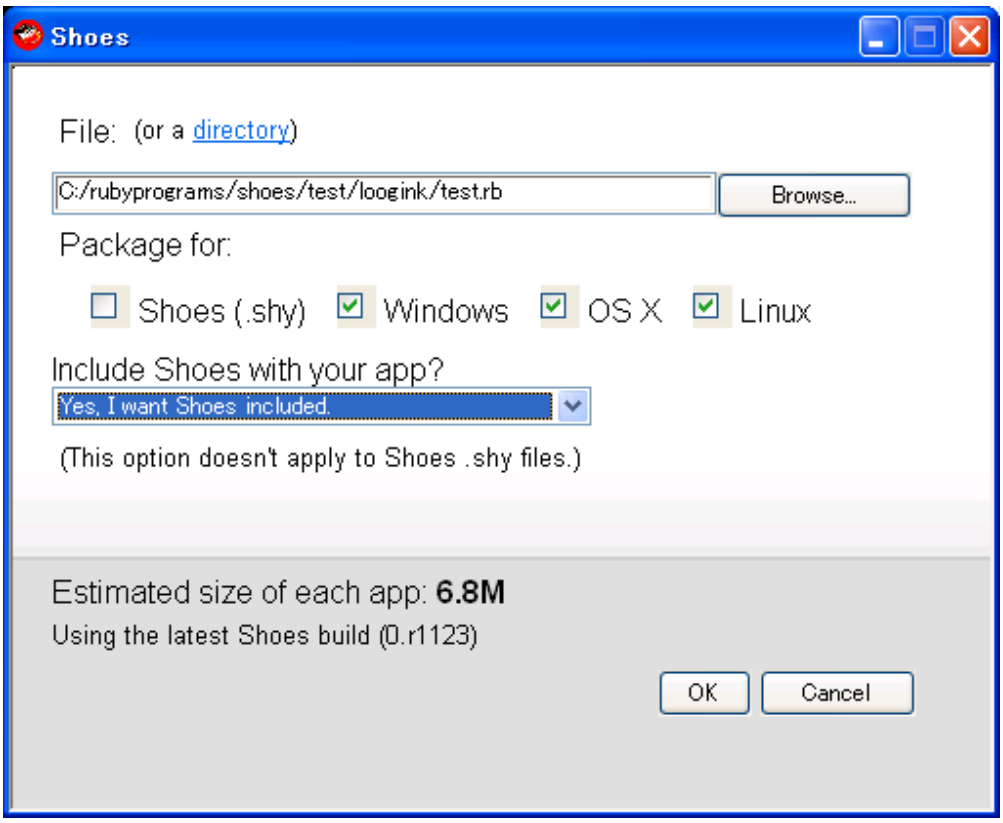


shoes_shy-1.png



After Shoes-0.r1057,the above message appears.
Because this feature is included in 'shoes --package'.

shoes_package.png

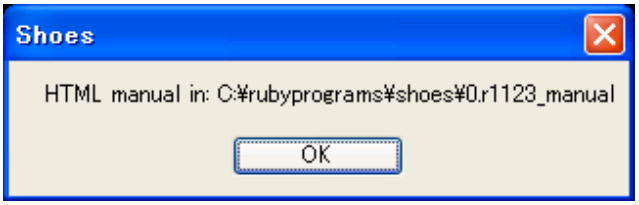


File select: sample1.rb
It works well. sample1.exe is created. And it can be executed well.

Directory select: code_wrapper_on_shoes_v0.2s
code_wrapper_on_shoes_v0.2s.exe is created. But it is not executed well. :(
Now still under construction...

Select shy: code_wrapper_on_shoes_v0.2s
code_wrapper_on_shoes_v0.2s.shy is created. It works very well!!

shoes_manual-html.png



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

It works well! Html files were created in my pc. Cool!

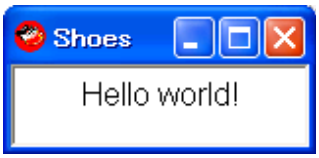
shoes_manual-html2.png

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 App 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 is the same as the above sample16.png.

```
# 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 is the same as the above sample16.png.

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

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

sample47.png is the same as the above sample16.png.

More information

Open question: when to use one coding style over another?
This question is provided by Jose Carlos Monteiro - Saturday, 15 November 2008, POIRPWSC101-1I

Year, good point!
I think there is no principles (no constraint), but I usually select like the following.

a) Sample 16 style
is most easy and simple. I like best.
So, I recommend you this style for the exercise 1 - day 2.

b) Sample 17 style
is a special style for using Shoes URLs.
Please refer to No. 9: Shoes.url

c) Sample 18 style
is a special style for using Shoes Widget class.
Please refer to the Widget class
And more information, but advanced article, is here

d) Sample 47 style
is a special style for using block or proc object.
Please refer to Open a new app window (Another example)

NOTE

Phew,... went directly into advanced topic....

The aboves are curious things. But I recommend you to learn Shoes step by step. Because Shoes is a developing project. Not so stable, hence there are many bugs and strang behaviors. If you stick on them, you will feel 'Shoes is not so fun'. I think it's not good. Shoes is fun!

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 Shoes programs:
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 sample19.rb

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

It really make a Shoes::Gang struct, not a Gang struct,
so change that line to this and it (sample19-1.rb) works well.

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

Modified code is

```
# 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

We can get mouse events. We can get a string from the system clipboard and also store a string into the 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

An example of the output.

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).
Then paste the clipboard text into the place you want.

*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...

after:
Creatures name list is:
cy
kamome
loogink
shaha
yar

We can get 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

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
```

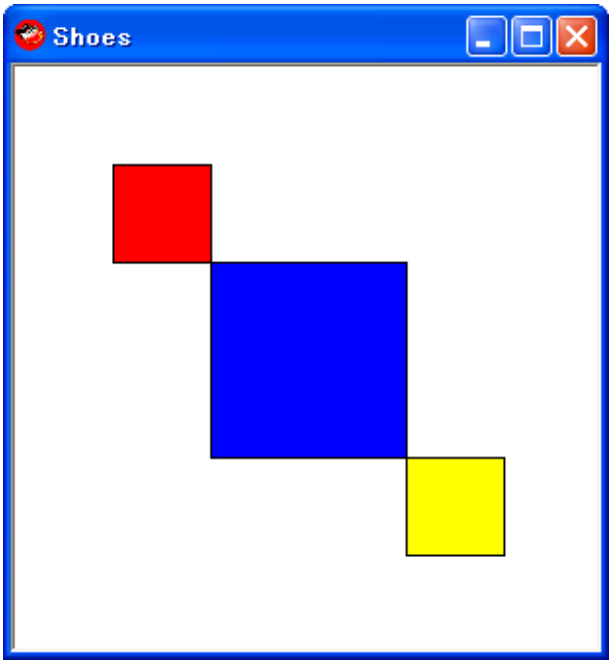
sample22.png

One more example

We can use :left and :top without definition in the custom class attributes. Because 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



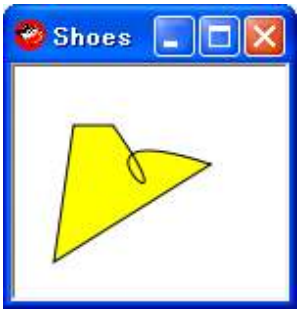
Original idea is provided by Emanue Carnevale, in the Shoes ML.

shape

We can make arbitrary shapes, beginning at 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



Oops, with Shoes-0.r925, it doesn't work well.
I'm not sure this behavior is the new spec or bug...
The above screenshot is with Shoes-0.r905.

sample23-1.png

With Shoes-0.r970, it works well. Although it is a little bit different than the above pic...
After Shoes-0.r1057, it works like Shoes-0.r970.

mask

We can use a masking layer. See the following information.

Cut Holes In Shoes And Get A Mask
http://hackety.org/2007/08/28/cutHolesInShoesAndGetAMask.html

```
# 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

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

sample24.png

Drawing directly onto images

We can add elements to images.
In the sample app below, Cy (green creature) has a star!

```
# 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

If you are using Shoes-0.r970, you need move the mouse off of the image once. After that it works well.
With Shoes-0.r1057, there is no need to do that.

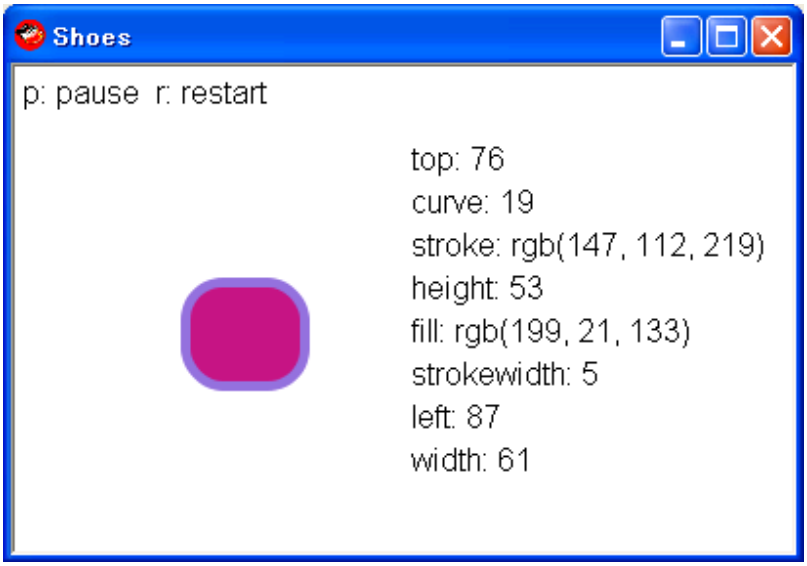
Style

We can change the style of the element with the style method. Calling the style method with no 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
```

sample26.png



We can style an entire class of elements at a time. :-D
But slot-by-slot would be impossible now. :(

```
# 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
http://hackety.org/2008/05/08/clearingUpTheWholeShoesAndRubyGemsDeal.html

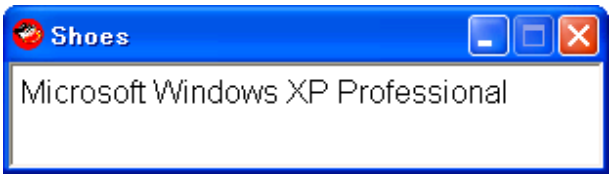
```
# sample27.rb
Shoes.setup do
  gem 'something'
end

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



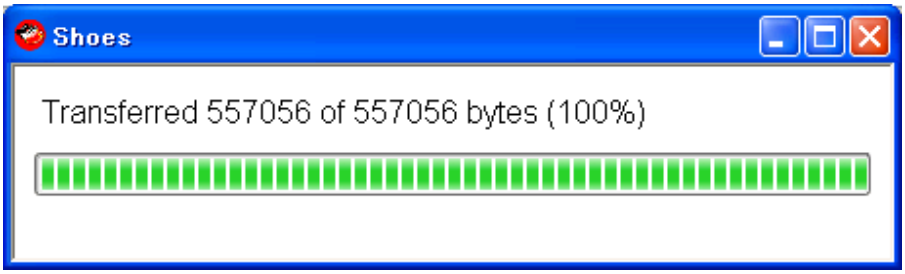
Original snippet was written by Massimiliano in the personal mail discussion.

Downloader

The methods download and progress are so cool.
Although percent and length methods don't work well now, transferred and fraction work well.

```
# 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



For more information about downloader, see the built-in manual.

Shoes includes the Hpricot library for parsing HTML.

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 revises the above sample11.rb.

```
# 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}. :)", :strok
    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
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

_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.

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.

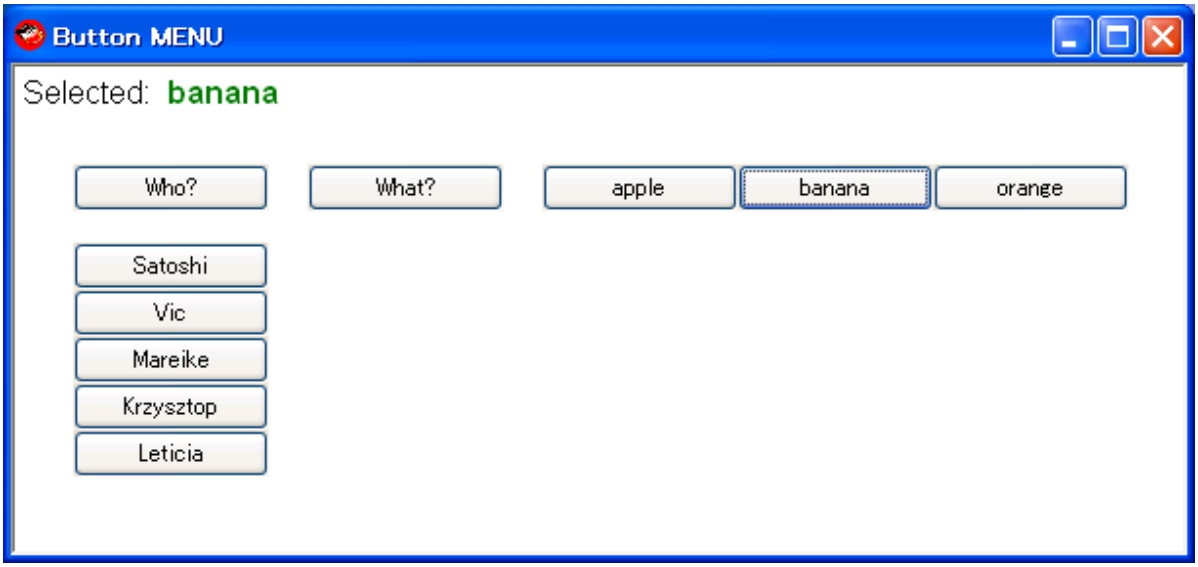
```
# sample32.rb
Shoes.app :width => 350, :height => 250, :title => 'Menus' do
  def menu items
    flow do
      items.each_with_index do |e, i|
        nostroke
        nofill
        b = image(:width => 100, :height => 21){rect(0, 0, 100, 21)}
        f = image(:width => 100, :height => 21){rect(0, 0, 100, 21, :fil
        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 Krzysztop 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
```

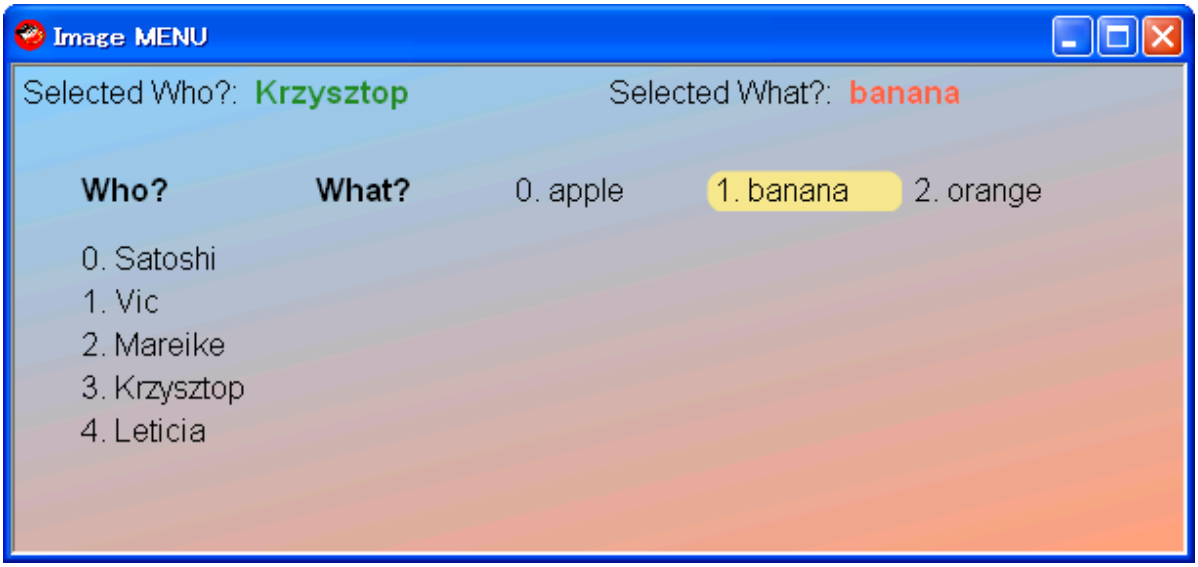
sample33.png



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
```

sample34.png



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

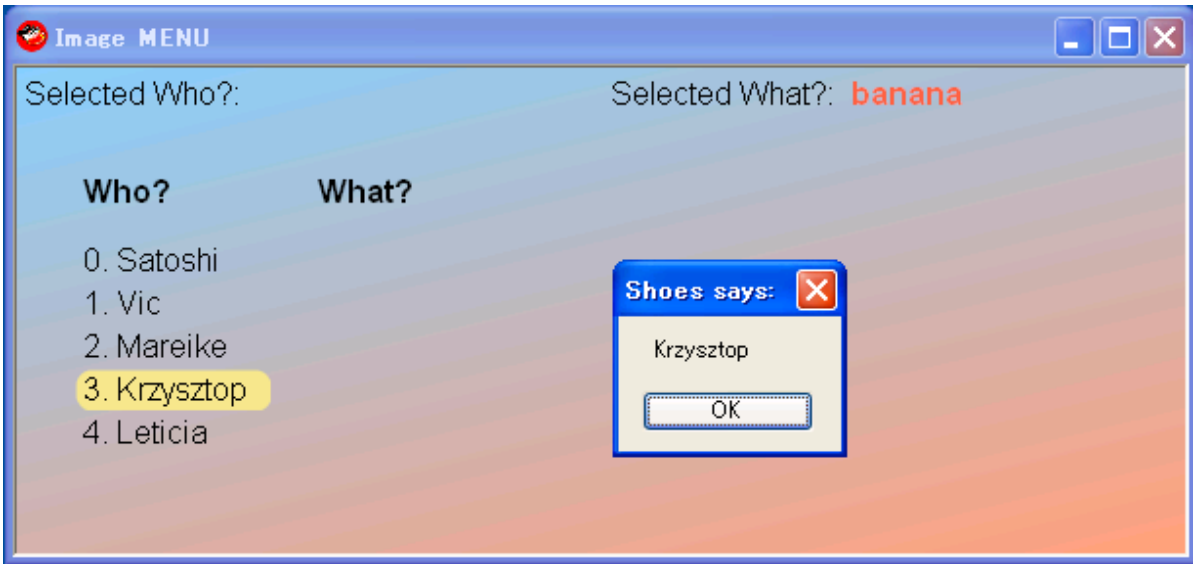
Line No. 20
b.click{@msg[n].text = strong(e)} ----> b.click{@msg[n].text = strong(e); @f[n].toggle}

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

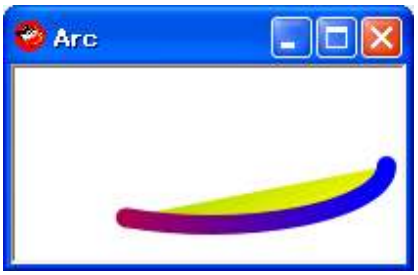
New arc and cap methods were released in the 970th build. See the following article:

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

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

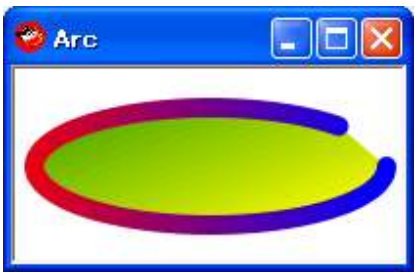
```
# sample35.rb
Shoes.app :width => 200, :height => 100, :title => 'Arc' do
  fill green.to_s..yellow.to_s, :angle => 45
  stroke red.to_s..blue.to_s, :angle => 90
  strokewidth 10
  cap :round

  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

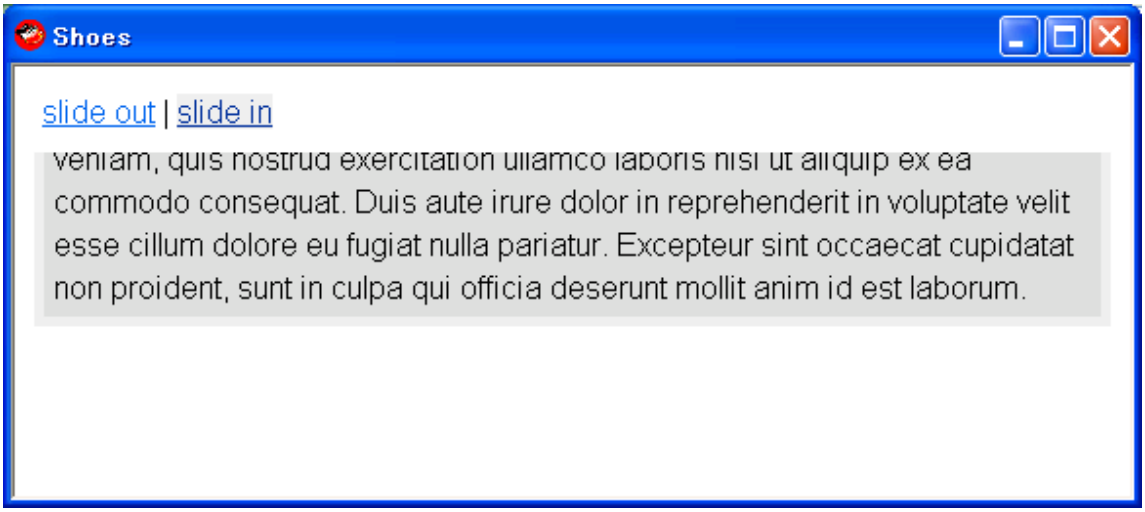
Click Yar and she will rotate (*1).

*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

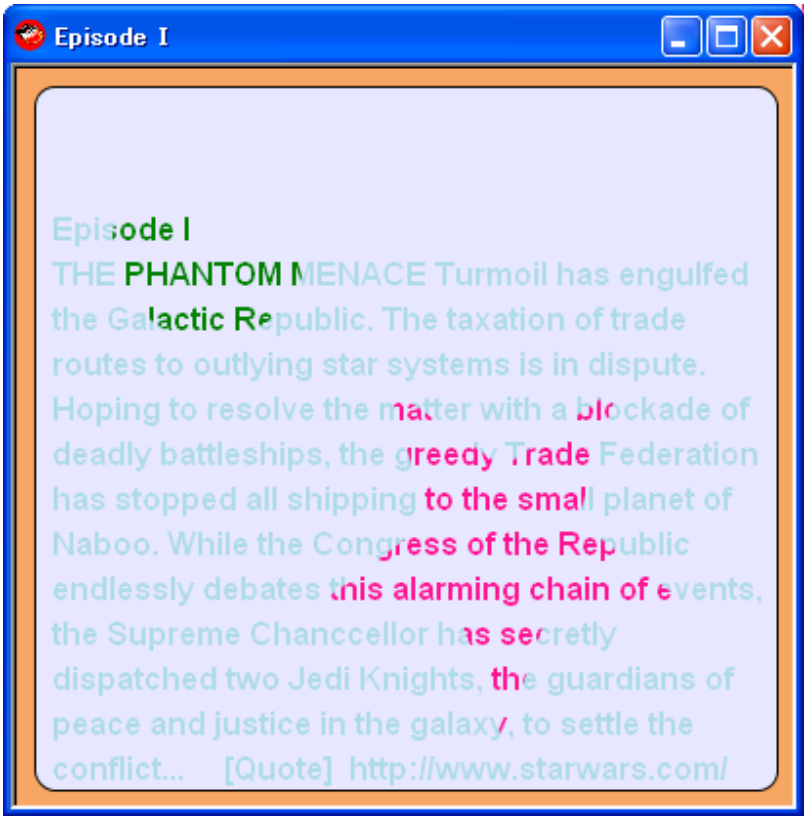


Next sample code, sample37.rb, which works almost similar behavior of the text message slide-in by using mask and animate methods.

```
# 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

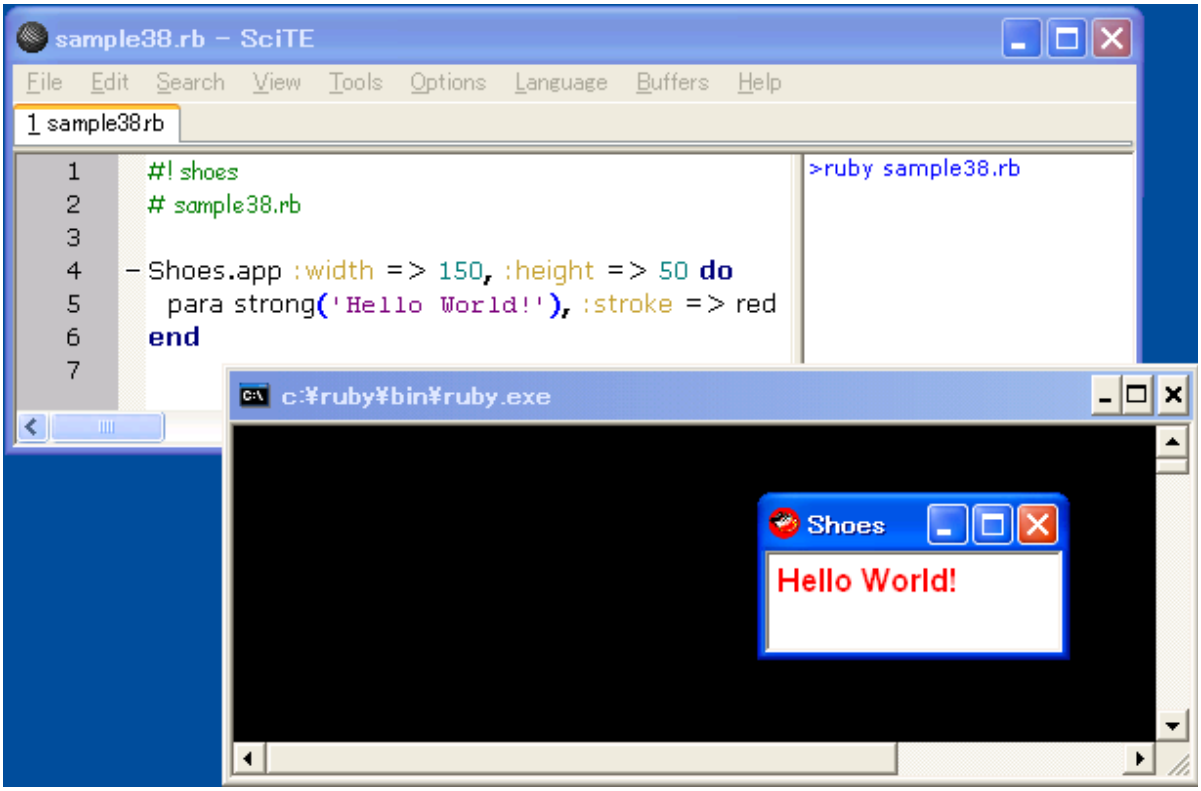


#! shoes

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?

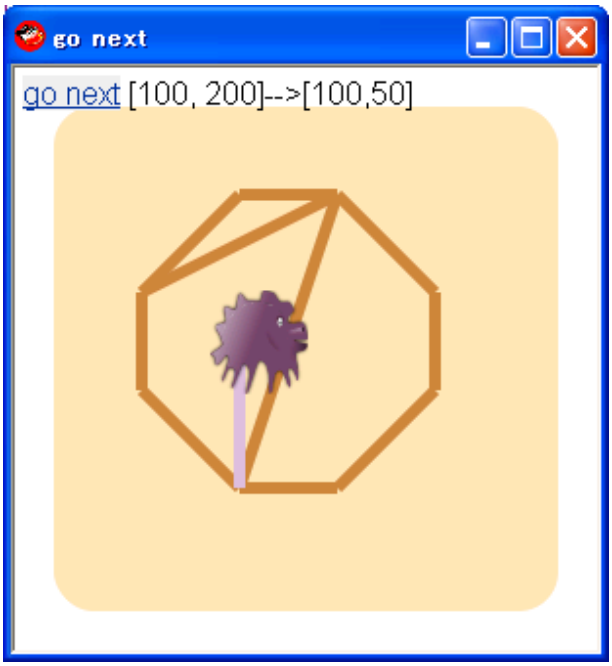
Sample39.rb has a require method to load the custom widget class stored in the other file (sample39-creature.rb).

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], [1
            [50, 100], [150, 50], [100, 200], [50, 100]]
  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
```

optional arguments

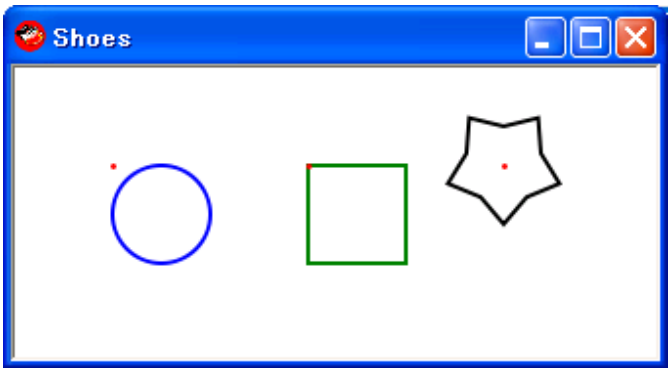
When we create an oval shape, like
oval :left => 50, :top => 50, :width => 30
The :left and :top positions are the top-left corner of the oval.

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

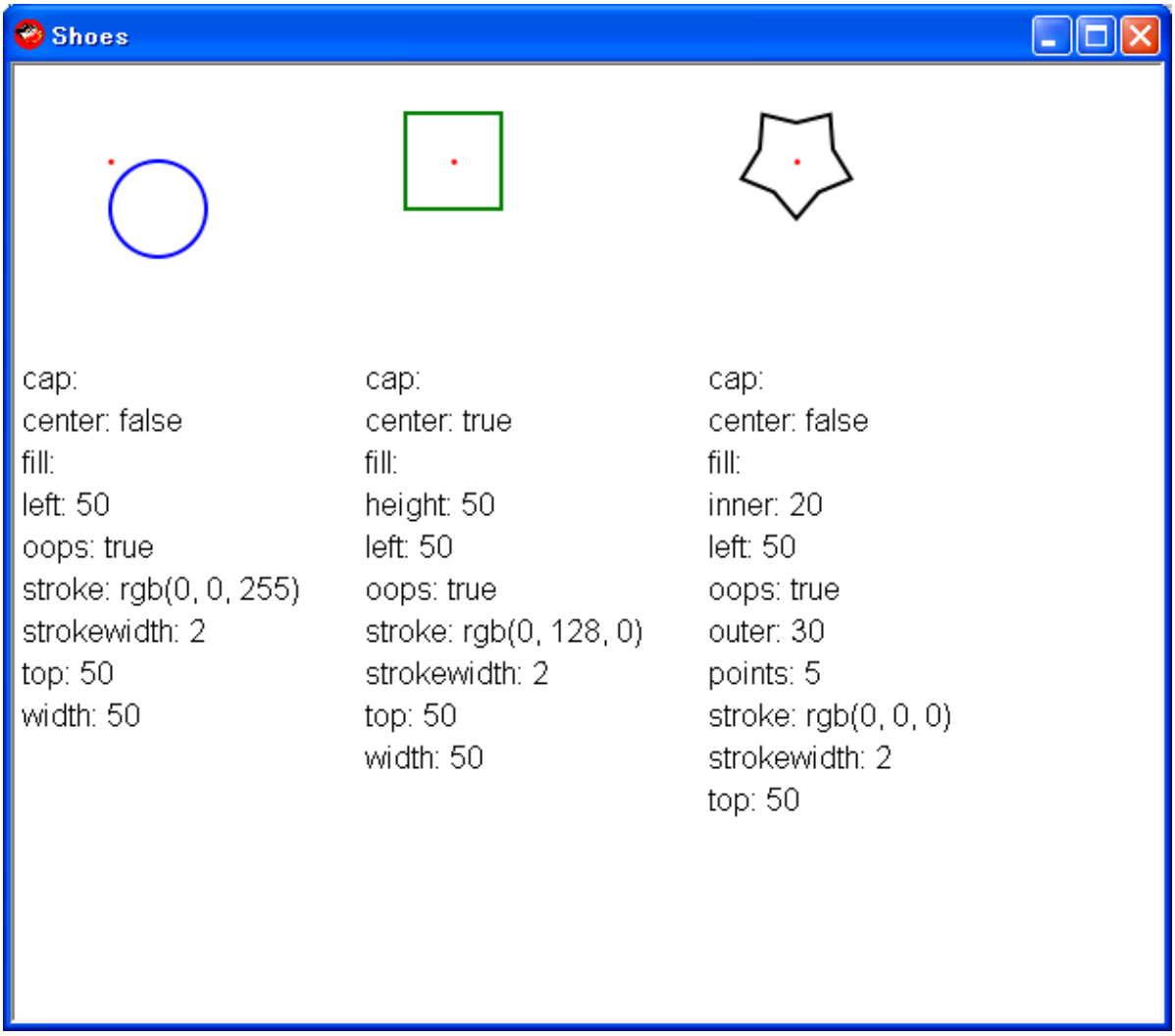


We can use the :center option to specify the coordinates. But it works well only in the case of the oval and rect, not the star method. If we add an undefined option like :oops as one of the arguments, no error will occur and nothing will happen, it will just be ignored.

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



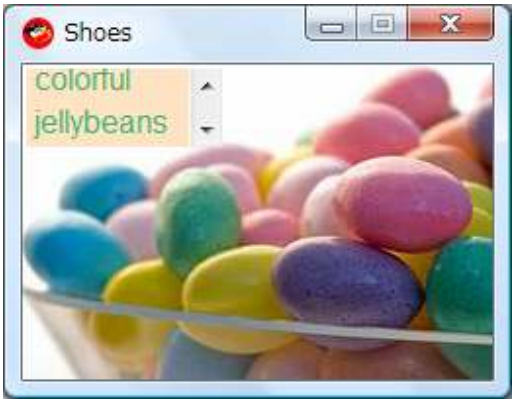
In the above sample40-1, the oval and rect methods accepted the :center option, but the star method ignored it as it ignored the undefined option :oops.

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
```

sample41.png

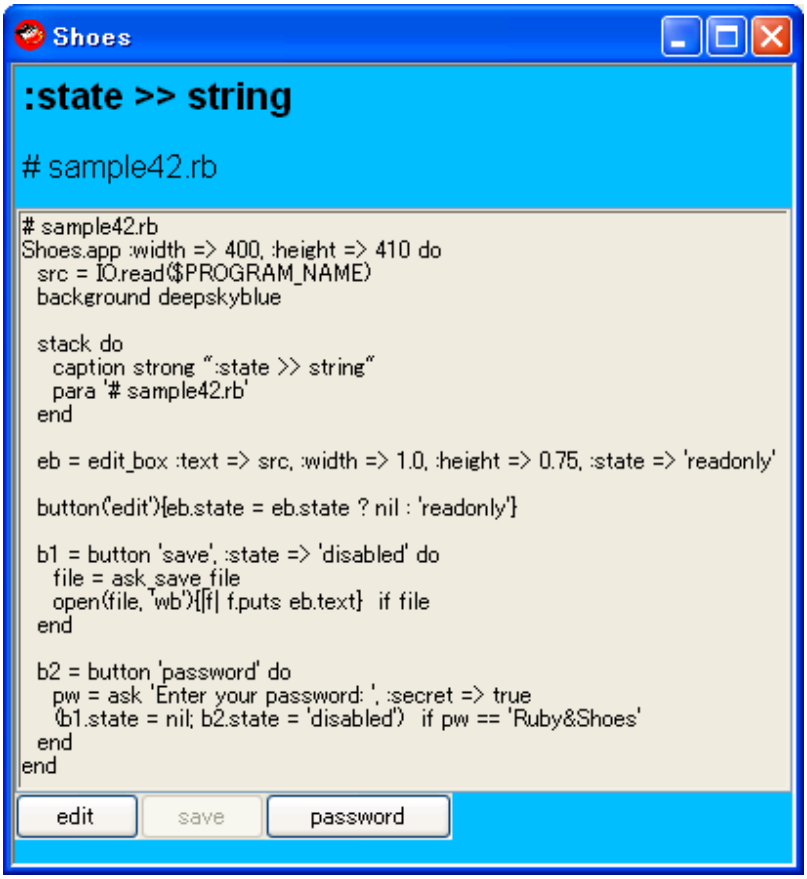


The :state style

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

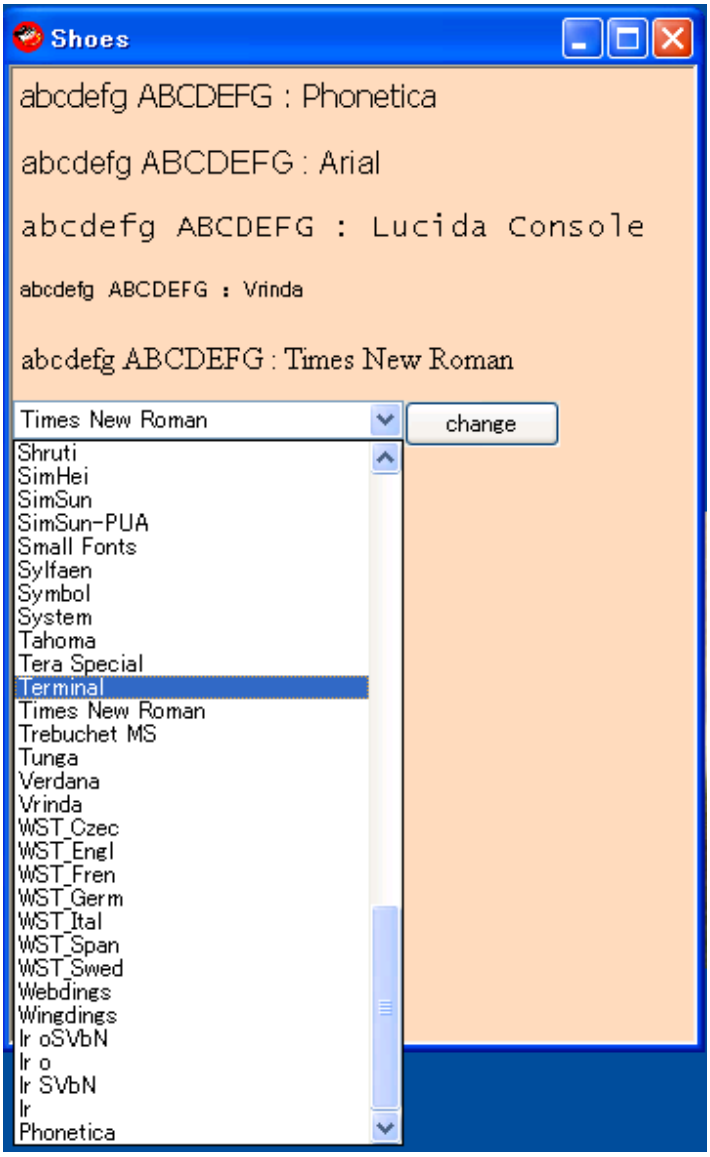
Shoes::FONTS is a complete list of the fonts you can use.
Loading from external fonts, such as TrueType and OTF files.

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

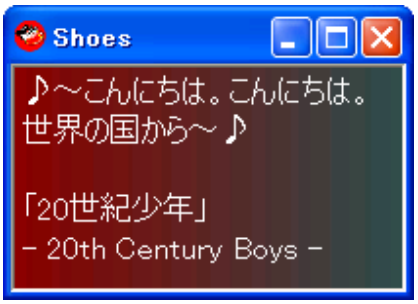
UTF-8

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

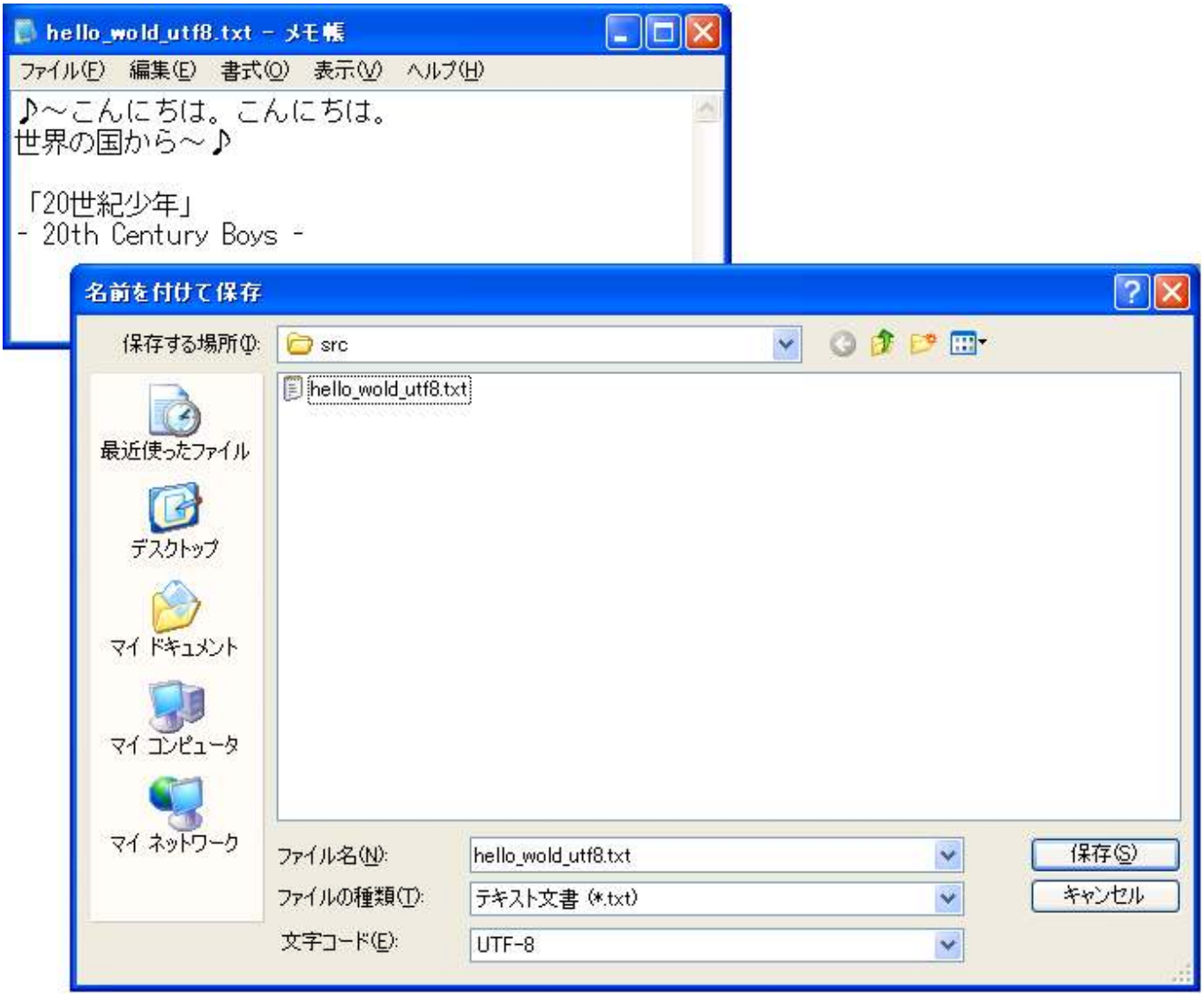
UTF-8 Everywhere
http://help.shoooes.net/Rules.html

```
# 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



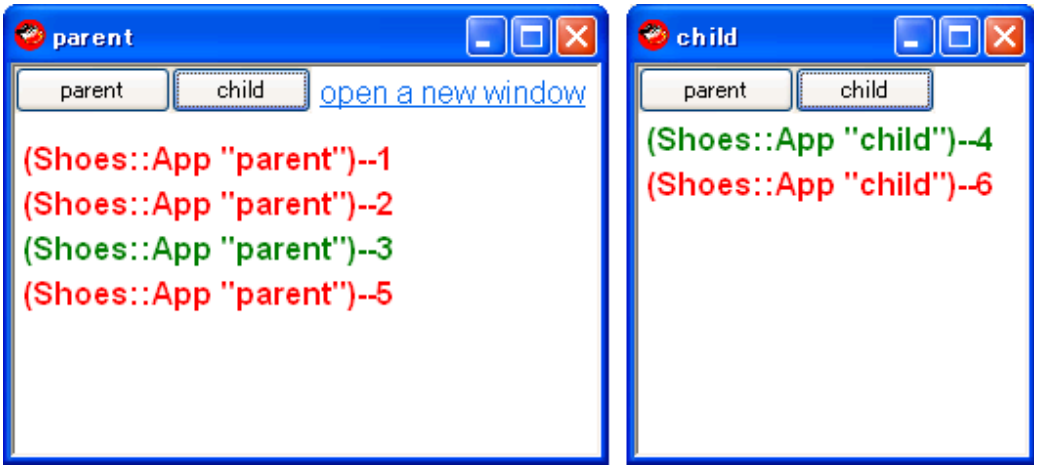
This Japanese text editor uses UTF-8.

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.

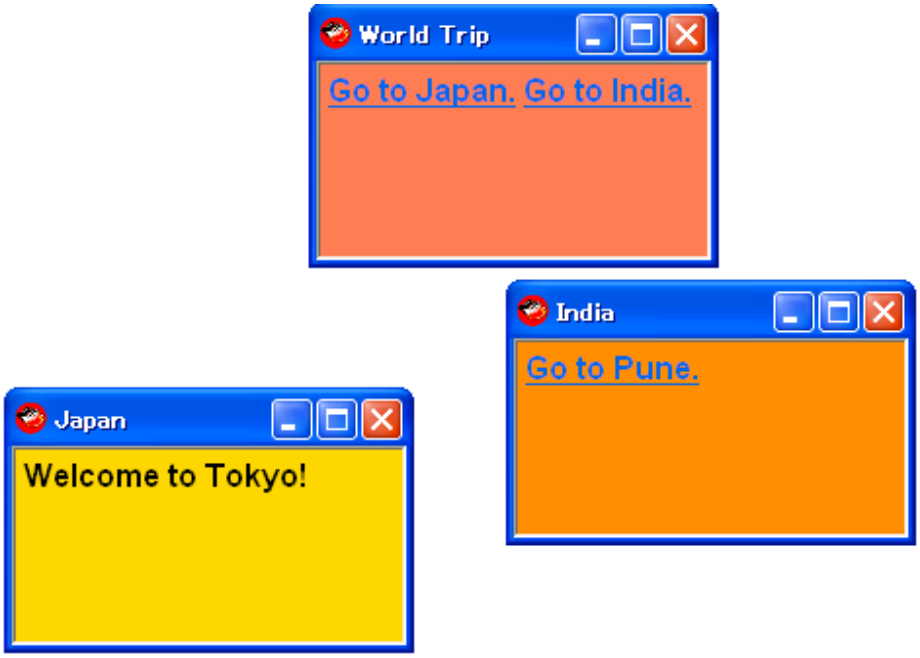
- open parent window.
- in parent window, click parent button, output is --1
- in parent window, click child button, output is --2
- click 'open a new window' link, open a child window.
- in child window, click parent button, output is --3
- in child window, click child button, output is --4
- in parent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

  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', chang 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.

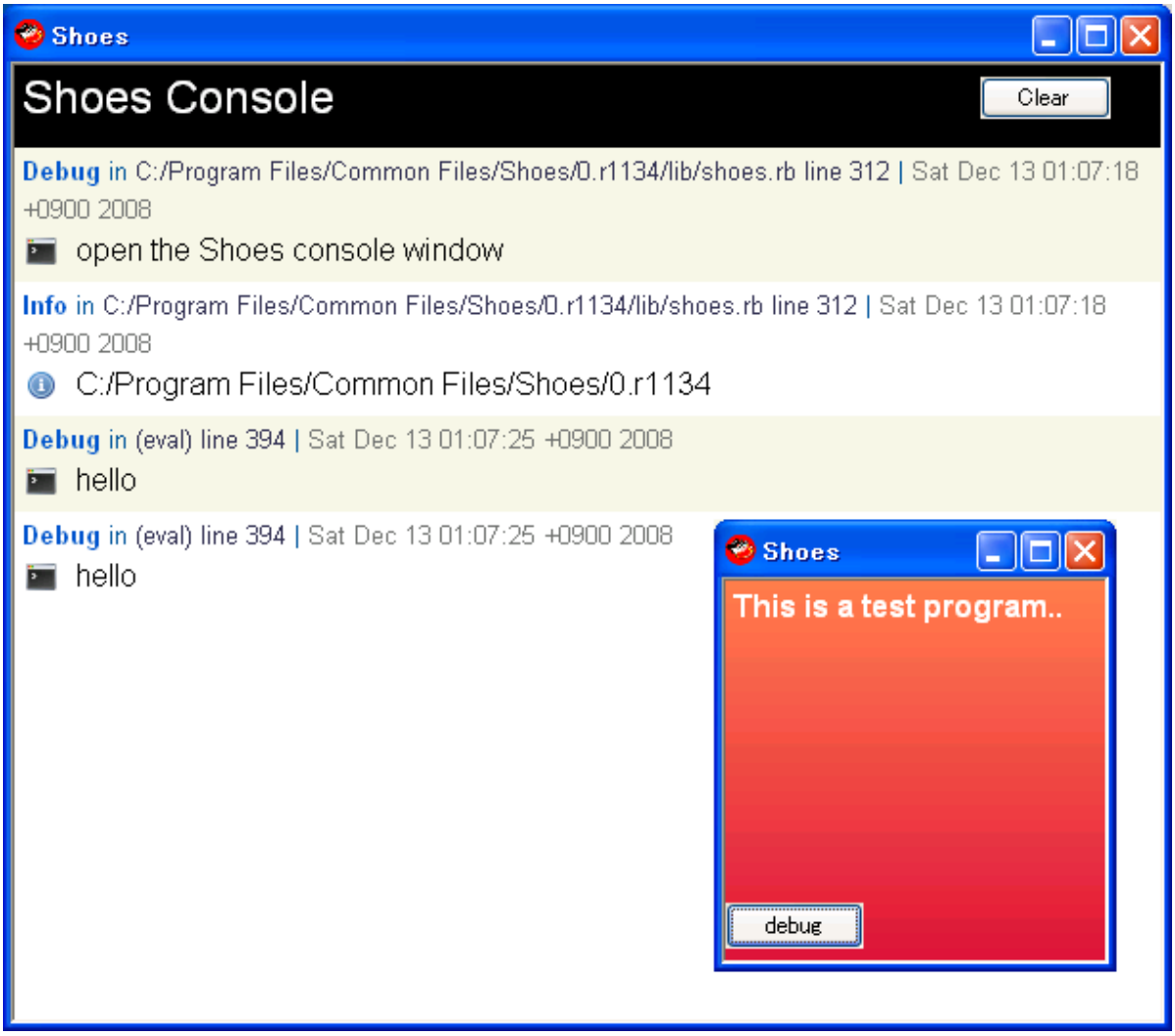
- click the button 'debug' twice on your app window

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[:le
      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)]}.pn
        blur_creature image(name, :name => name, :top => 20 + i * 60, :l
      end
    end
  end
end
```

sample54.png

Note

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

Video playback

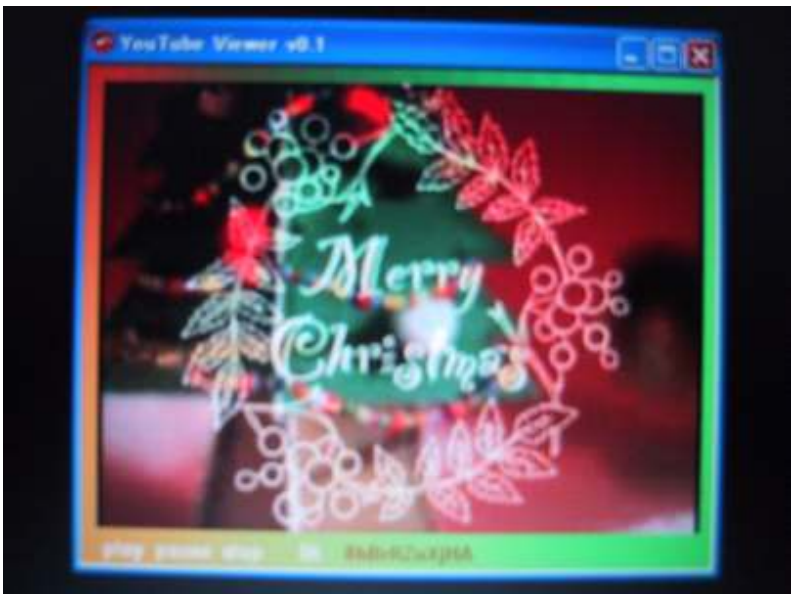
We can do playback YouTube videos on Shoes!

I referred to the following web site.

GUIfy 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=\
OEgsToPDskKhuXfknhF5eWPOvhe-nw5P", :autoplay => true}
```

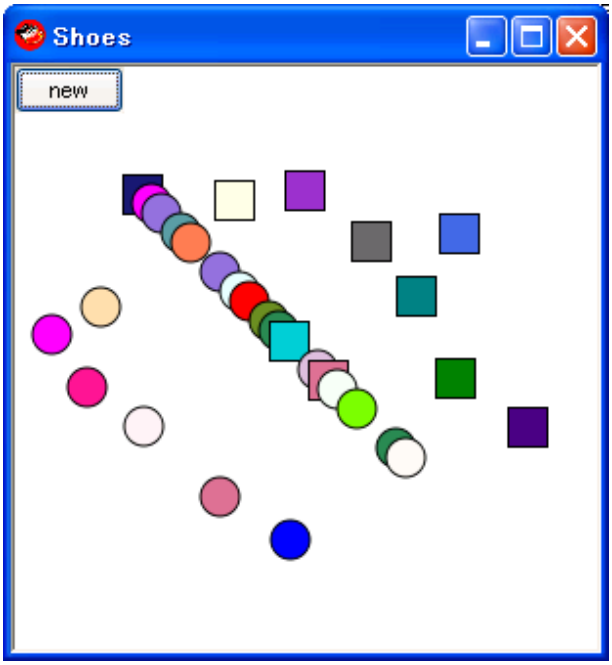
Just one-liner solution playing YouTube video on Shoes!

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

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

Exercises

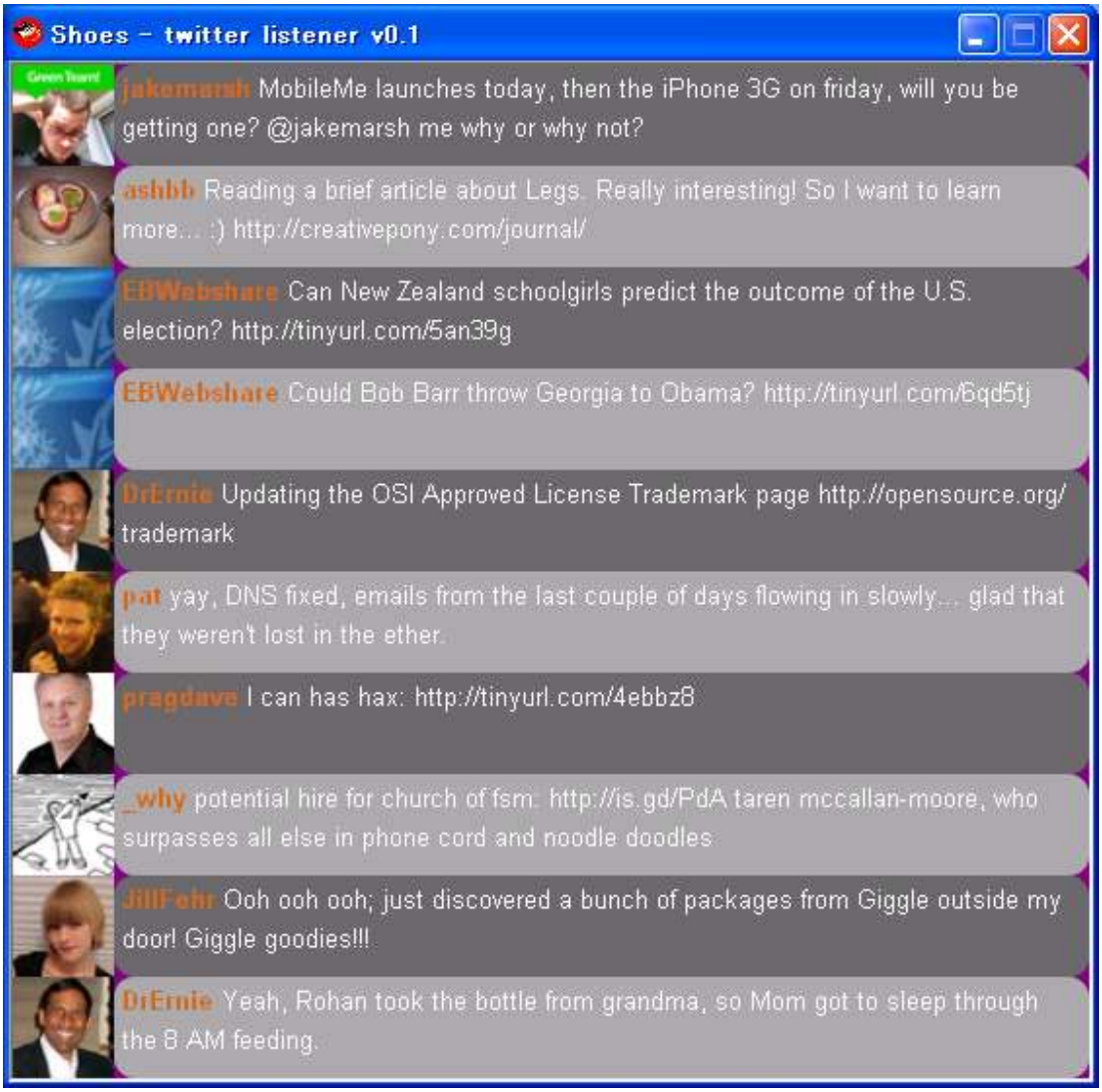
Exercise 1 twitter client (reader)

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!

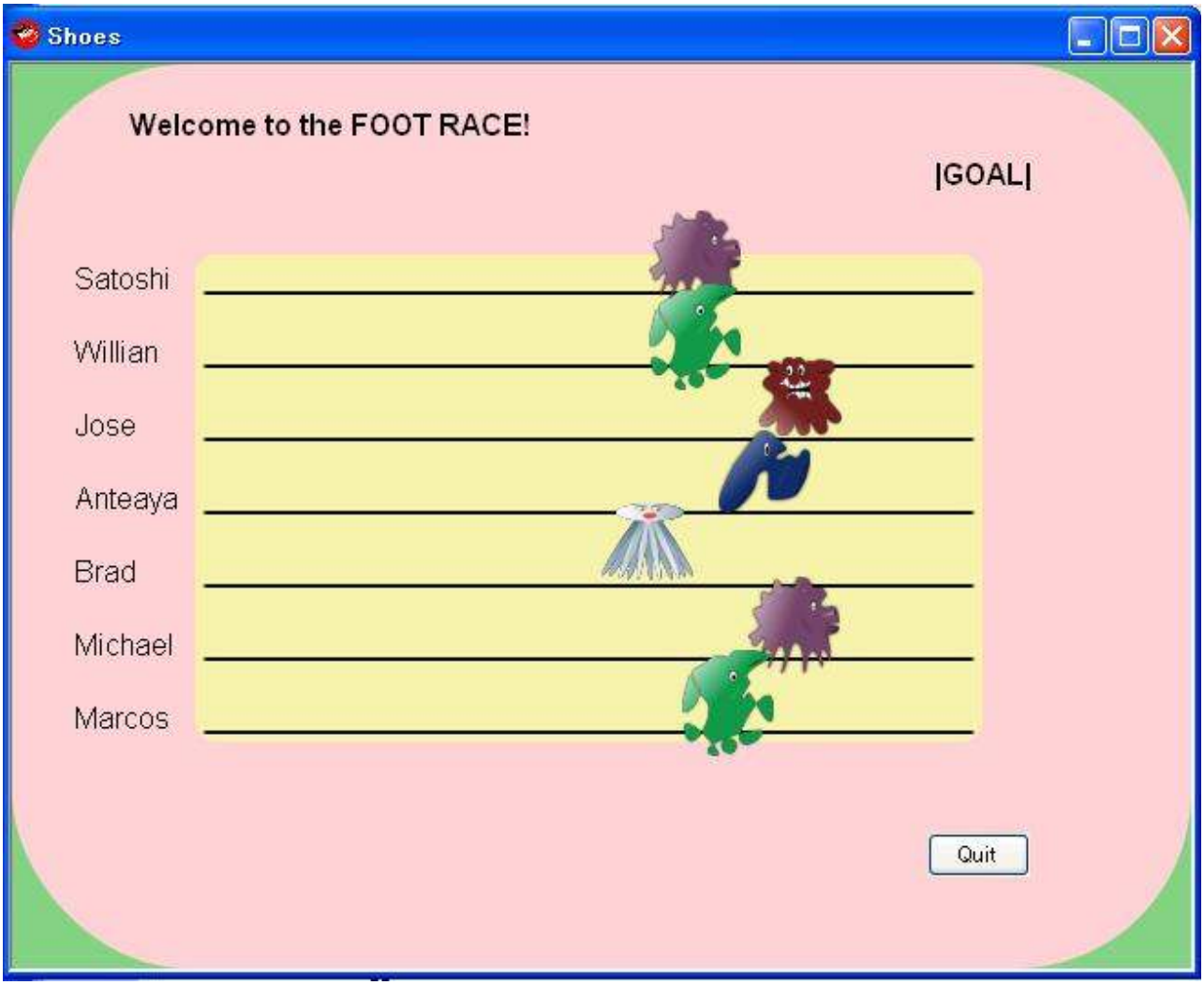
Exercise 2 footracer

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



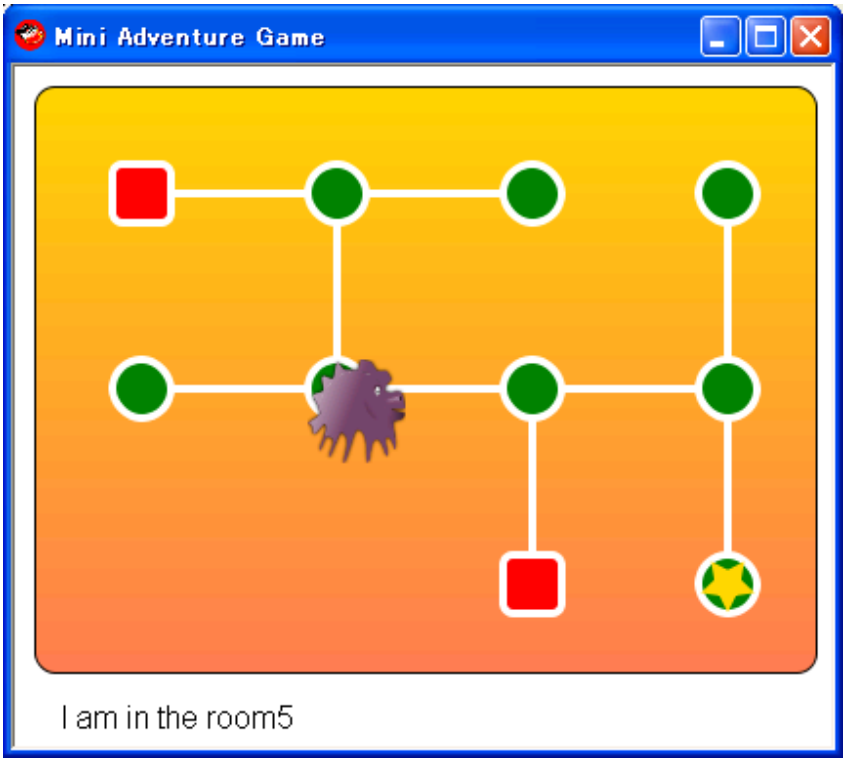
Have fun!

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 roon 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
```

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:nsw
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
```

Have fun!

Assignment 3 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, :alig
      @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

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)

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

Under consideration... :)

memo

- 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

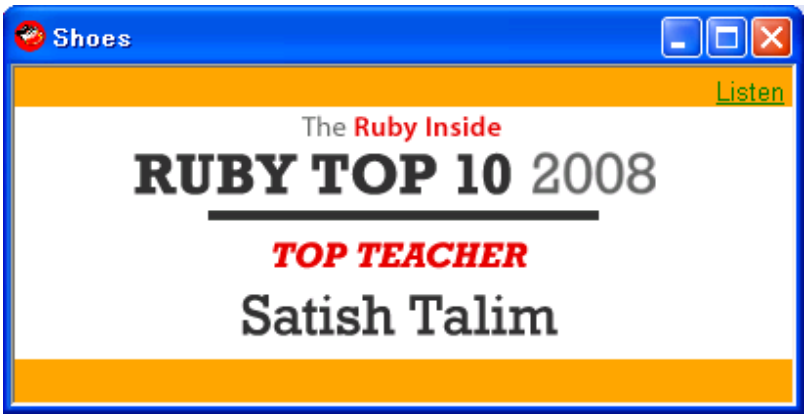
Gallery No.1

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 'shell.png', :curve => 5
    @line = para '', :stroke => white, :weight => 'bold'
  end

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

    @line.text = @line.text[0..-2] + "\n" + k  if @line.text.to_a.last.d
  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 POIRPWSC101-2l

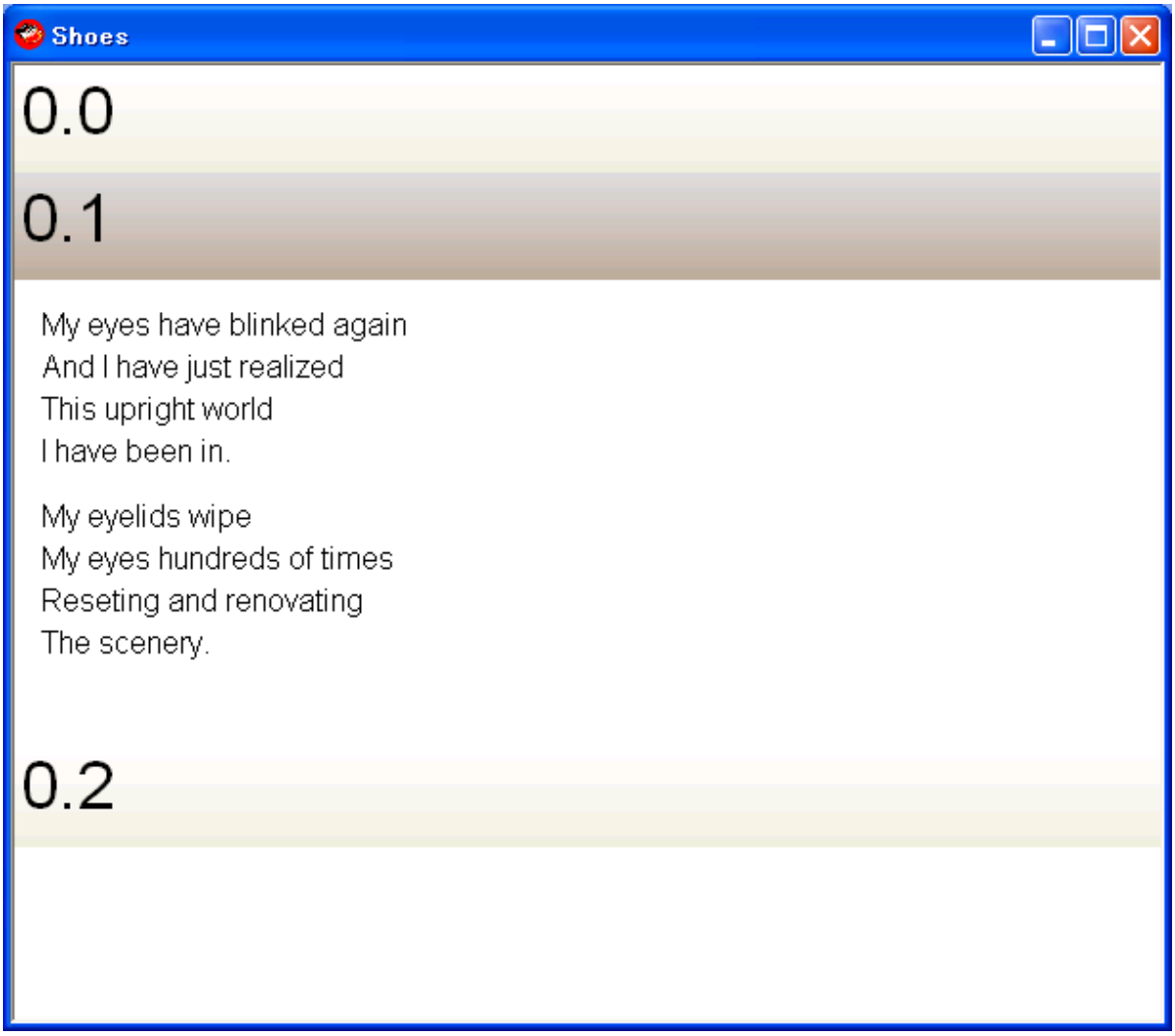
```
# gallery3.rb
Shoes.app :title => "Live Code", :width => 500, :height => 240, :resizab
  background purple..white

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

Built-in Samples

simple-accordion

simple-accordion.png


Shoes
0.0
0.1
My eyes have blinked again
And I have just realized
This upright world
I have been in.
My eyelids wipe
My eyes hundreds of times
Reseting and renovating
The scenery.
0.2

```
# 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

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).

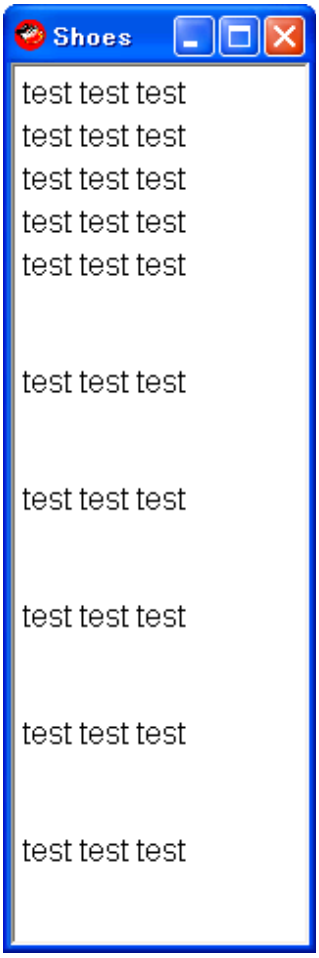
#9:
If you don't need to open the first page as a default, just write like this:

```
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).

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

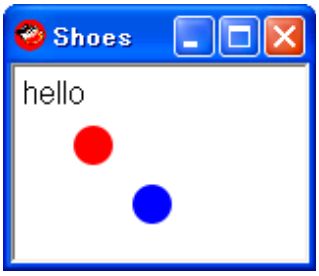
Try to comment out :height => 30 and run.
The list_box doesn't show the items.
This strange behavior occurs only on Windows. On Mac OS X, it doesn't.

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://shoooes.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://newwws.shoooes.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 Creaturs' 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
```

sample93.png

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

FIN.