

Shoes Tutorial

- For the Shoes App Rookie Creators -

September 22, 2008

by Satoshi Asakawa

Table of contents

1. Introduction.....	2
2. Download Shoes	2
3. First step	2
4. Birds-eye view (Survey basic features)	2
4.1 Concept.....	2
4.2 No.1 para	3
4.3 No.2&3 stack and flow	4
4.4 No.4 button.....	6
4.5 No.5 image.....	7
4.6 No.6 edit_line	8
4.7 No.7 link	9
4.8 No.8 background	10
4.9 No.9 Shoes.url	11
4.10 No.10 clear.....	12
5. Tips for creating our original Shoes apps	14
5.1 Open Shoes built-in manual and Shoes console window	14
5.2 Output messages on the Shoes console window	15
5.3 shoes --help	16
5.4 App object and coding style.....	19
5.5 Built-in Constants and methods.....	20
5.6 Scope: A tip of using the YAML file	20
5.7 keypress, mouse and clipboard	22
5.8 the Widget class	24
5.9 shape.....	25
5.10 mask.....	26
5.11 Drawing directly on to images	27
5.12 Style.....	28
5.13 Shoes.setup	29
5.14 Downloader	30
5.15 Assign Shoes URL dynamically.....	31
5.16 Classes List and Colors List	32
5.17 start, stop and restart	34
5.18 Combination of image objects show/hide and mouse hover/leave.....	35
5.19 arc and cap	39
5.20 widget with block.....	40
6. Assignment.....	41
6.1 Assignment 1 – twitter client (reader)	41
6.2 Assignment 2 – footracer	42
7. Relevant web sites (Links)	43
8. Appendix.....	43
8.1 Advanced article	43
8.2 Shoes mailing list in English	43
8.3 Shoes mailing list in Spanish	43
8.4 Shoes IRC channel.....	43
9. Trivia.....	44

1. Introduction

Shoes is a cross-platform tiny graphics and windowing toolkit for the Ruby programming language written by [_why](http://en.wikipedia.org/wiki/Why_the_lucky_stiff). (http://en.wikipedia.org/wiki/Why_the_lucky_stiff)

All sample programs and data files in this tutorial are able to download from [here](http://www.rin-shun.com/rubylearning/xxxxxxx.zip). (<http://www.rin-shun.com/rubylearning/xxxxxxx.zip>)

Some sample programs are the same in the NKS (The first public manual of Shoes. See chapter 7.)

2. Download Shoes

Download Shoes from [this web site](http://shoooes.net/downloads/) and pick [the installer](http://help.shoooes.net/Introducing.html).

(<http://shoooes.net/downloads/>)

(<http://help.shoooes.net/Introducing.html>)

We use the latest revision, Shoes-0.r970, in this tutorial.

3. First step

There is [a tutorial](http://shoooes.net/tutorial/) written by _why.

(<http://shoooes.net/tutorial/>)

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.

4. Birds-eye view (Survey basic features)

4.1 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 can be simulated with images.

There are ten essential methods to get going with Shoes.

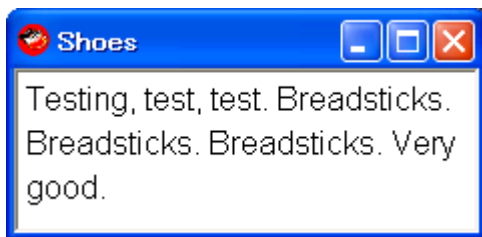
4.2 No.1 para

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

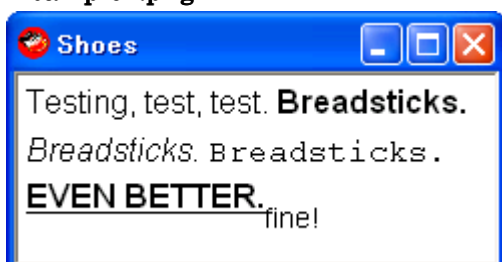


strong	: bold
em (emphasized)	: italics
code	: monospaced font
ins (inserted)	: single underline
sub (subscript)	: lowering the text by 10 pixels, x-small font

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



4.3 No.2&3 stack and flow

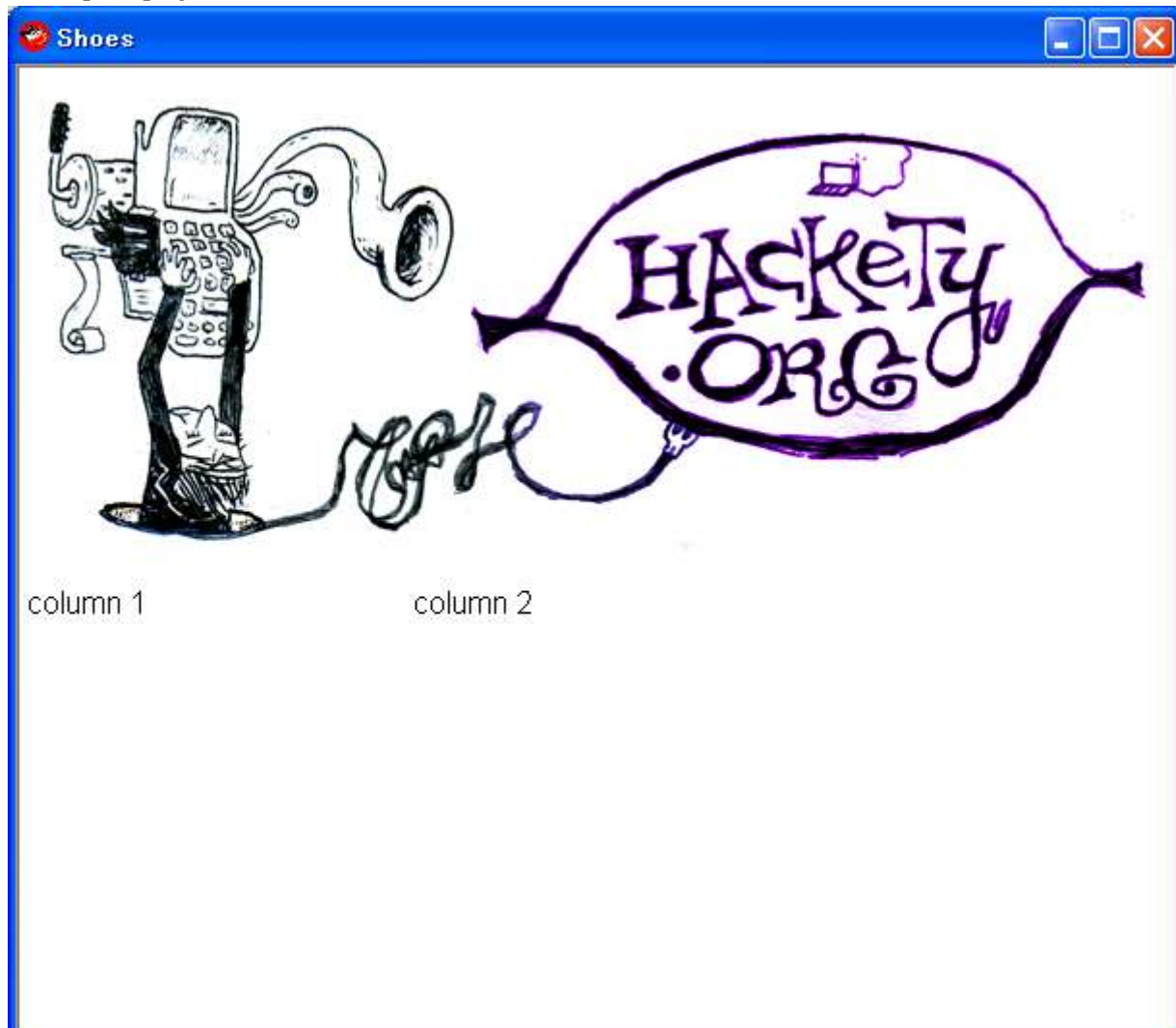
At first, read the following web page:

<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 need to correct the path of 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



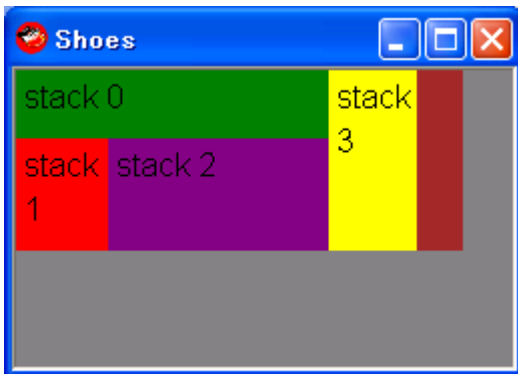
More complex sample code is:

```
# 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
```

sample4.png



4.4 No.4 button

`button("Press Me")`

which creates a new button and

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

how the block fires when clicked and

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

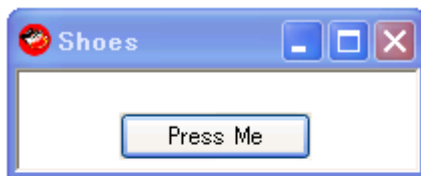
will place the button at coordinates (50, 20).

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

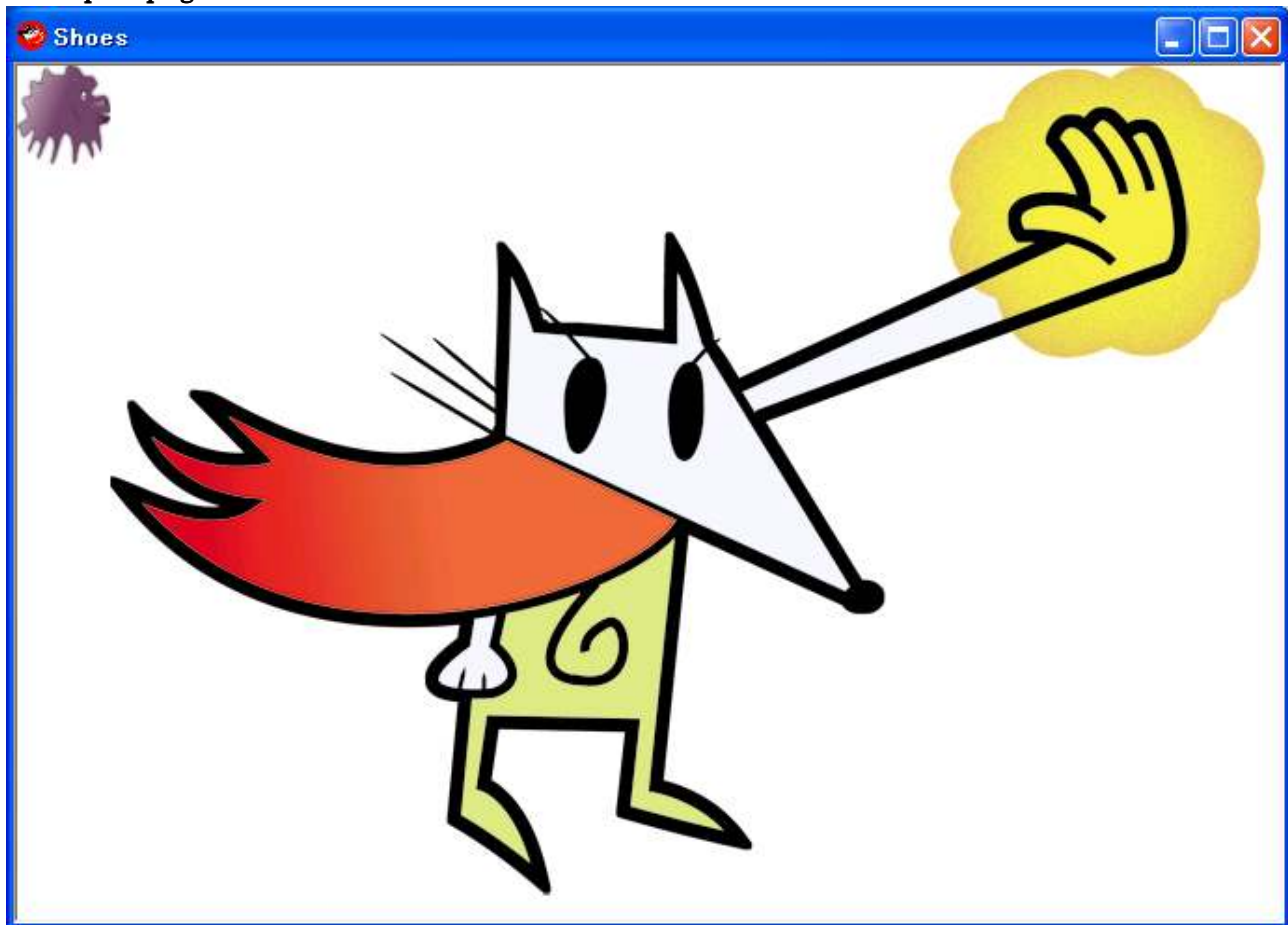


4.5 No.5 image

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

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

sample6.png



4.6 No.6 edit_line

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



4.7 No.7 link

Hyperlinks. We have three way 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 (Dir.pwd + '/loogink.png'), :click => "http://shoooes.net/"
end
```

sample8.png



4.8 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 Dir.pwd + '/loogink.png', :strokewidth => 15
end
```

sample9.png



In NKS(Nobody Knows Shoes), 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



4.9 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 view point, just seems like a page of 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 Dir.pwd + '/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 Dir.pwd + '/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



4.10 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 Dir.pwd + '/' + name + '.png', :left => 30, :top => 10
  end

  random_creatures

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

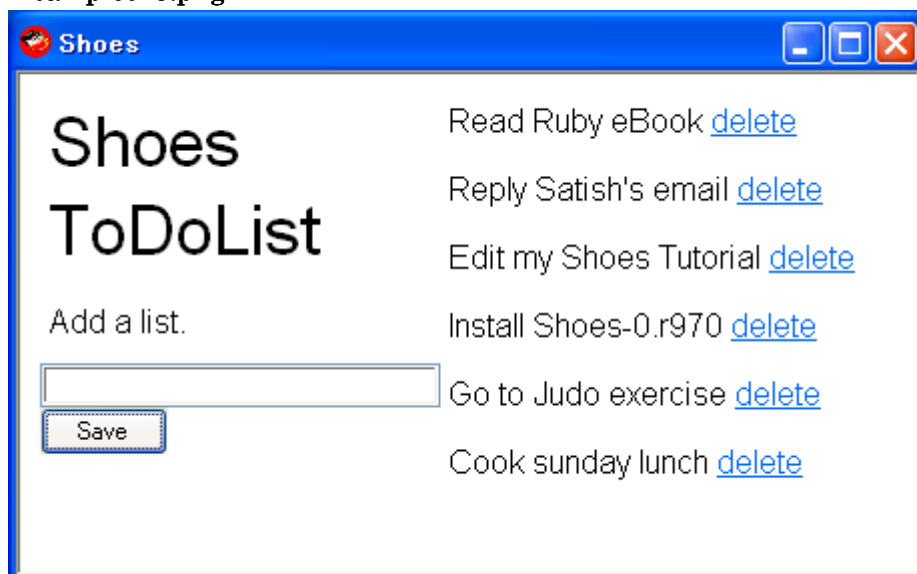
sample12.png



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
```

samples13.png



5. Tips for creating our original Shoes apps

5.1 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).

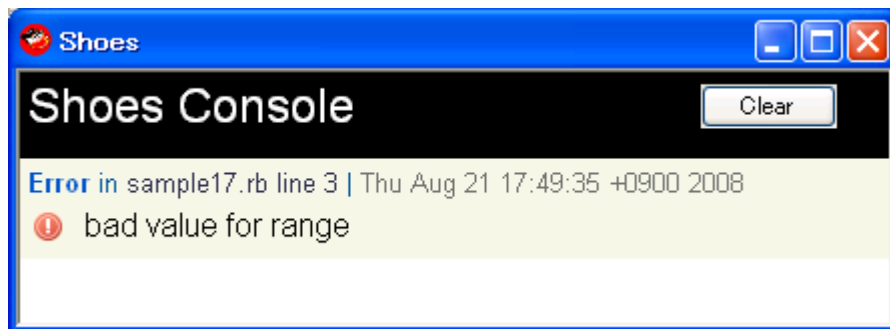
```
shoes -m  
or  
shoes --manual
```

Or type `Alt + ?` on any Shoes app window.
Or select from the menu. See [here](http://shoooes.net/manuals/).
(<http://shoooes.net/manuals/>)

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

```
# sample14.rb  
Shoes.app do  
  background blue..red  
end
```

shoes_console.png

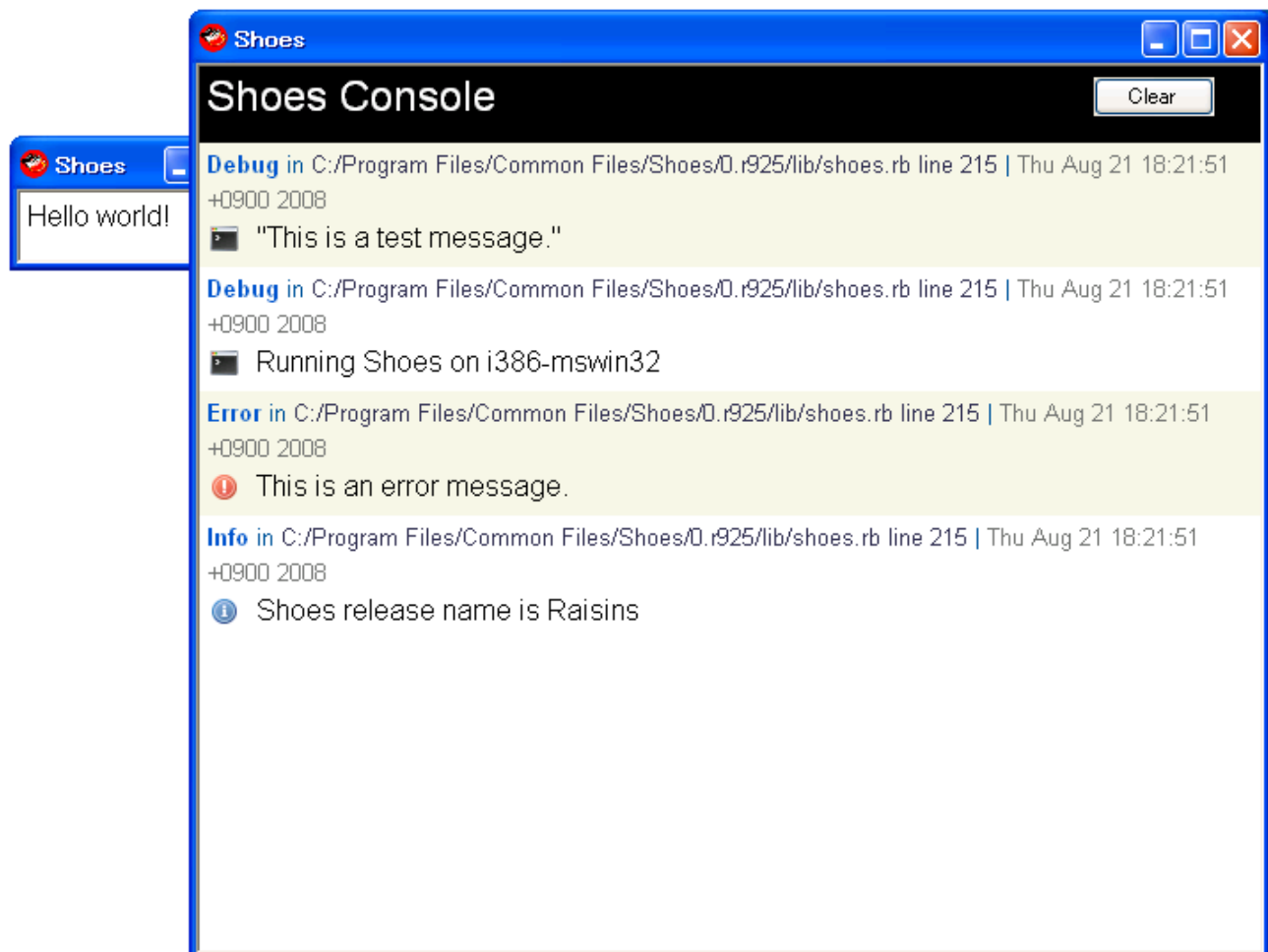


5.2 Output messages on the Shoes console window

We can put the message on 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



5.3 shoes --help

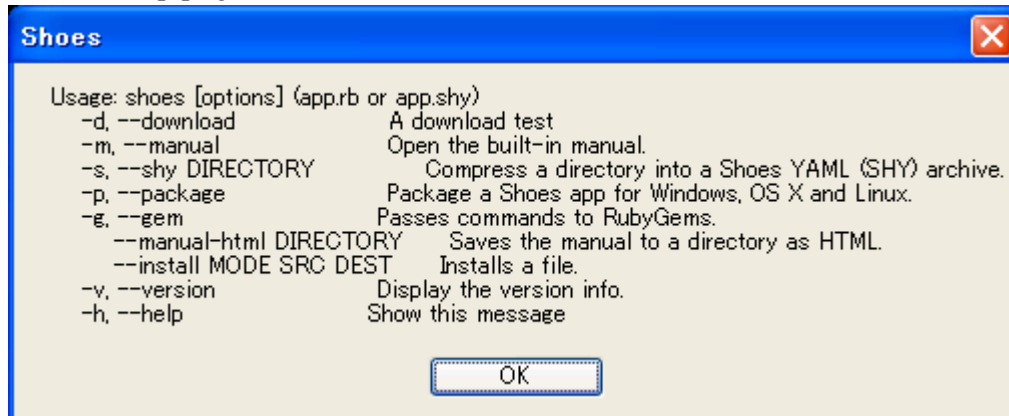
Type the following on your pc console (terminal window).

```
shoes -h
```

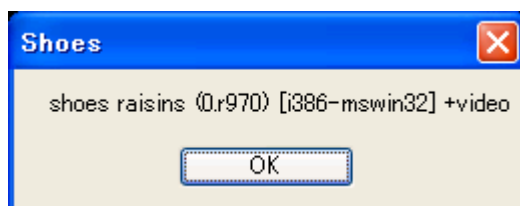
or

```
shoes --help
```

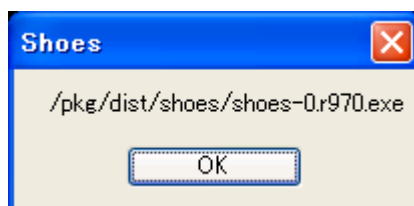
shoes_help.png



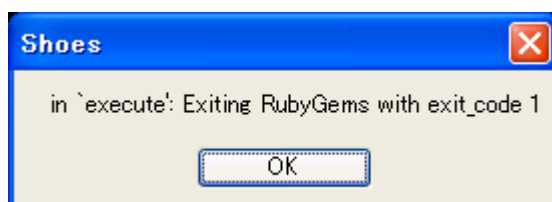
shoes_version.png



shoes_download_test.png



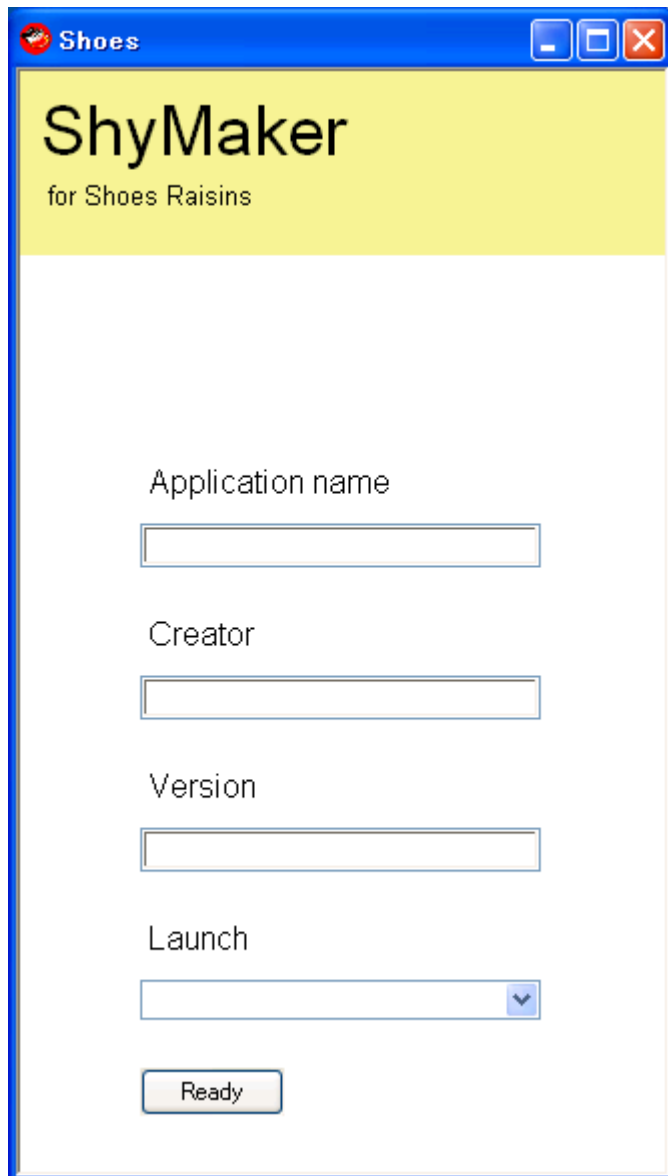
shoes_gem.png



```
shoes -g install hpricot
```

Oops, an error will be happen. Perhaps now under construction...

shoes_shy.png



The image shows a screenshot of a Windows-style application window titled "Shoes". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area has a yellow header with the text "ShyMaker" in a large, bold font, and "for Shoes Raisins" in a smaller font below it. The main area is white and contains four input fields with labels: "Application name", "Creator", "Version", and "Launch". Each label is positioned above its corresponding input field. The "Launch" field is a dropdown menu with a small downward arrow on the right. At the bottom of the form is a button labeled "Ready".

Shoes

ShyMaker

for Shoes Raisins

Application name

Creator

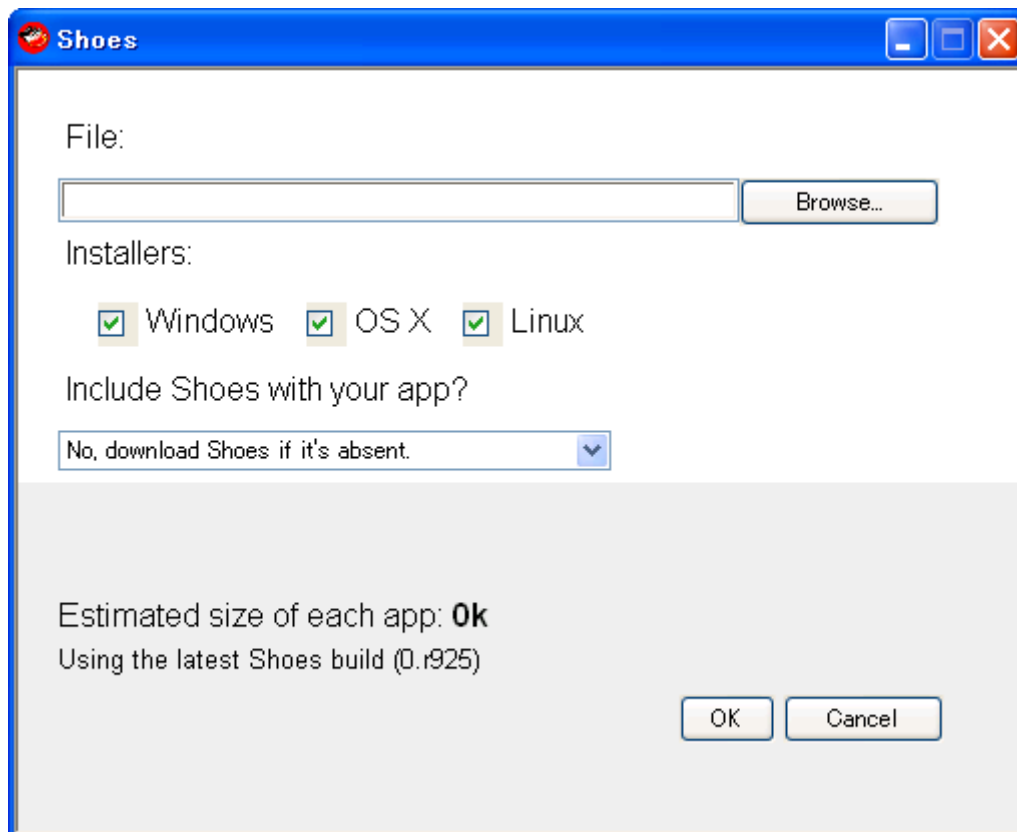
Version

Launch

Ready

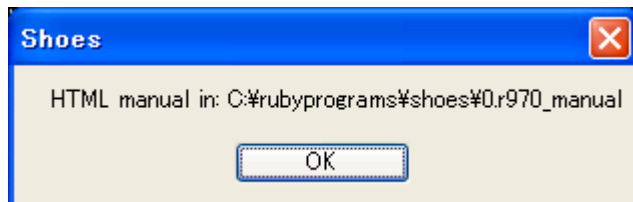
Open the app window, but it doesn't work well. Now under construction...

shoes_package.png



Open the app window, but it doesn't work well. Now under construction...

shoes_manual.html.png

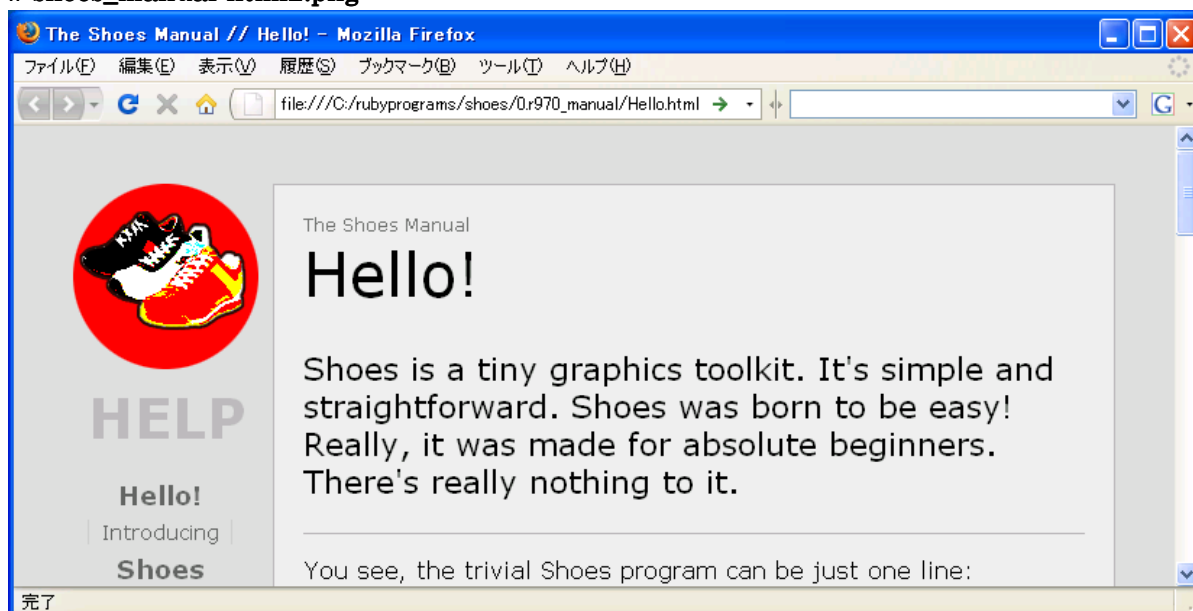


```
C:\>cd C:\Program Files\Common Files\Shoes\0.970
```

```
C:\Program Files\Common Files\Shoes\0.970>shoes --manual-html C:\rubyprograms\shoes\0.970_manual
```

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

shoes_manual.html2.png

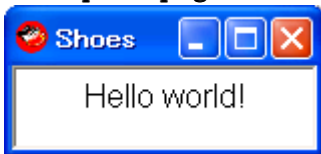


5.4 App object and coding style

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. The Shoes program has three coding style:

```
# 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 Hello < 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.

5.5 Built-in Constants and methods

Built-in Constants:

Shoes::RELEASE_NAME

Shoes::RELEASE_ID

Shoes::REVISION

Built-in methods:

These methods can be used anywhere throughout Shoes programs:

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

Read the Built-in manual -> Hello! -> Built-in section.

5.6 Scope: A tip of using the YAML file

```
# make_sample19_yaml.rb
require 'yaml'

data =<<-EOS
Satoshi Asakawa, Japan
Tom Jonson, Italy
EOS

Gang = Struct.new :name, :country

gangs = []
data.each{|d| gangs << Gang.new( *(d.chomp.split(',') ) )}

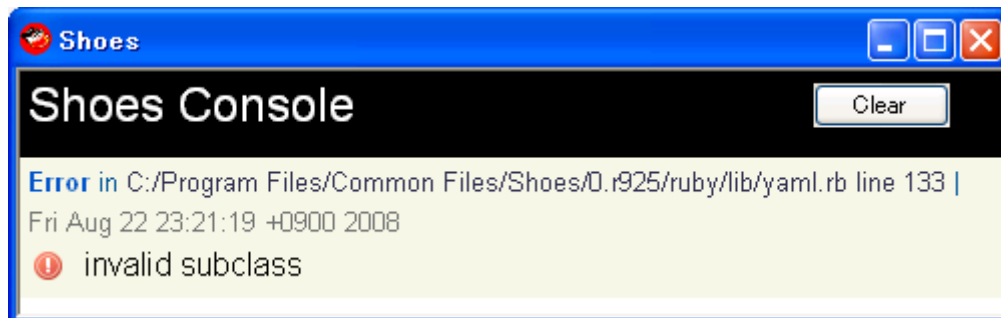
open('gangs.yml', 'w'){|f| f.puts YAML.dump(gangs)}
```

```
# gangs.yml
---
- !ruby/struct:Gang
  name: Satoshi Asakawa
  country: " Japan"
- !ruby/struct:Gang
  name: Tom Jonson
  country: " Italy"
```

```
# sample19.rb
require 'yaml'

Shoes.app 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 sample19.rb

```
Gang = Struct.new :name, :country
```

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

```
::Gang = Struct.new :name, :country
```

```
# 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



5.7 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(Dir.pwd + '/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.remove 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 expected. But with Shoes-0.r970, Yar rotates when mouse moves out of the Shoes window. This behavior is a bug. It will be fixed in the next Shoes release.

after:

Creatures name list is:

cy
kamome
loogink
shaha
yar

We can get keypress.

```
# sample21.rb
```

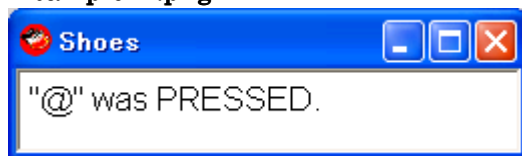
```
Shoes.app :width => 250, :height => 40 do
```

```
  @info = para 'NO KEY is PRESSED.'
```

```
  keypress{|key| @info.text = "#{key.inspect} was PRESSED."}
```

```
end
```

sample21.png



5.8 the Widget class

A custom Shoes widget is setup by inheriting from the Widget class.

And Shoes then creates a method using the lowercased name of the class which is used in your app.

```
# sample22.rb
class Answer < widget
  attr_reader :mark
  def initialize word
    para word
    @mark = image(Dir.pwd + '/loogink.png', :width => 20, :height => 20).hide
  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

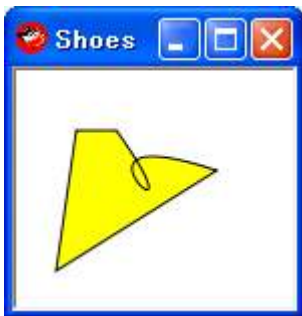


5.9 shape

We can make the arbitrary shape what ever we want.
Beginning at coordinates (left, top).

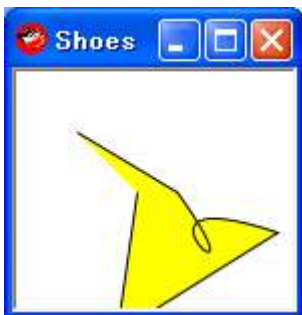
```
# sample23.rb
Shoes.app :width => 140, :height => 120 do
  fill yellow
  shape :left => 30, :top => 30 do
    line_to 50, 30
    curve_to 100, 100, 10, 20, 100, 50
    line_to 20, 100
    line_to 30, 30
  end
end
```

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 be a little bit different from the above pic...

5.10 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)

(<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



5.11 Drawing directly on to images

We can draw some elements on to the images.

In the below sample app, 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 Dir.pwd + '/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.remove
    msg.replace ''
  end
end
```

sample25.png



This pic is with Shoes-0.r925.

Oops, doesn't work with 0.r970... But it will be fixed in the next Shoes release.

5.12 Style

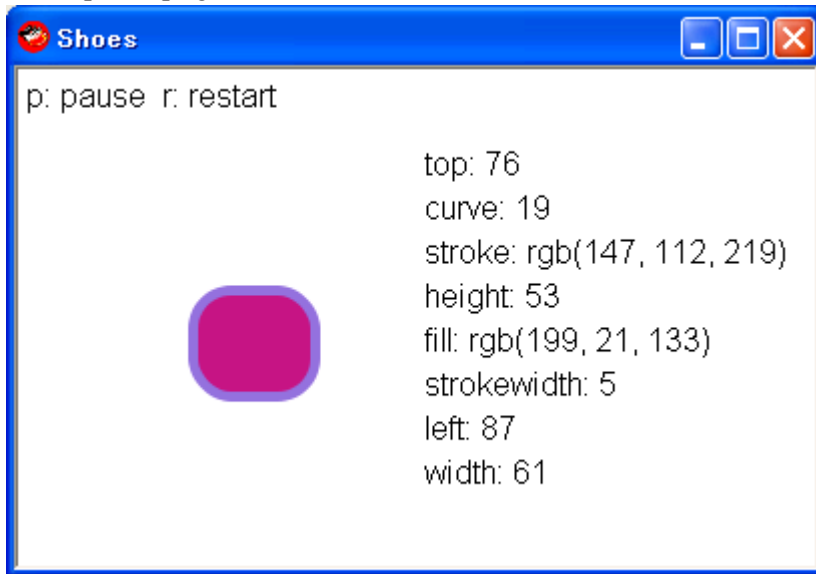
We can change the style of the element with the `style` method. Calling the `style` method with no arguments returns a hash of the styles presently applied to the element.

```
# 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.remove
    when 'r'
      @a.remove if @a
      clear{sampling}
    else
    end
  end
end
```

sample26.png



5.13 Shoes.setup

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

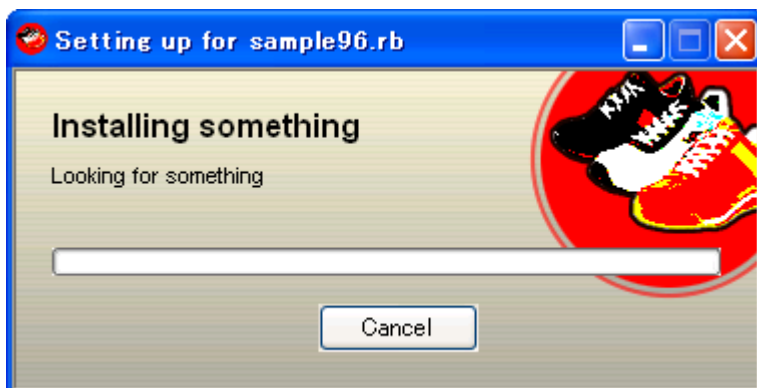
[Clearing Up The Whole Shoes And RubyGems Deal](http://hackety.org/2008/05/08/clearingUpTheWholeShoesAndRubyGemsDeal.html)

(<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



5.14 Downloader

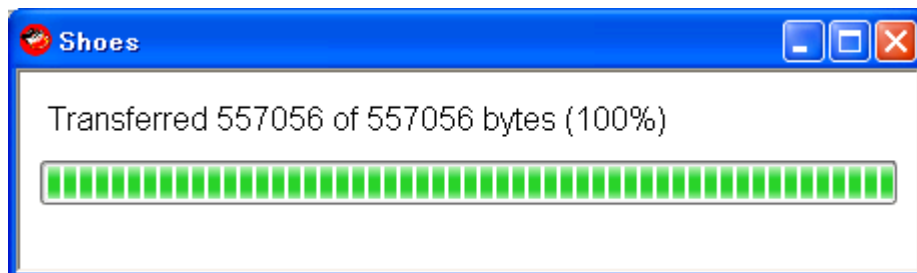
The methods `download` and `progress` are so cool.

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://shooooes.net/dist/shoes-0.r970.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} bytes
        (#{dl.percent}%)\"
        p.fraction = dl.percent * 0.01},
      :finish => proc{|dl| status.text = 'Download finished'},
      :error => proc{|dl, err| status.text = "Error: #{err}" }
    end
  end
end
```

sample28.png



More information about downloader. See the built-in manual.

Shoes includes the Hpricot library for parsing HTML.

5.15 Assign Shoes URL dynamically

We can use regular expressions to assign Shoes URL dynamically.
Shoes passes the match data to the method as the argument.
Show the following sample code revised 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 Dir.pwd + '/' + @@c[n].name + '.png', :left => 70, :top => 10
    para "¥n" * 3
    para strong "#{@@c[n].sex} is #{@@c[n].name.capitalize. :})", :stroke => white
    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.

5.16 Classes List and Colors List

We can see the colors list in the built-in manual.

But we will also be able to see them by 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.

5.17 start, stop and restart

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

```
#sample31.rb
Shoes.app :width => 150, :height => 70 do
  def number_on_disk
    fill eval(@color)
    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('chang') do
    @slot.clear
    @color = %w(green red blue yellow)[rand(4)]
    @i = 0
    @slot.append{number_on_disk}
  end
end
```

sample31.png



5.18 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 => yellow,:curve => 8)}.hide

        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



This one is another sample code. It shows the 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 => 100){@msg.text = strong(e)}}}
      else
        @f[n].clear
        @msg.text = ''
      end
      @toggle[n] = !@toggle[n]
    end
  end

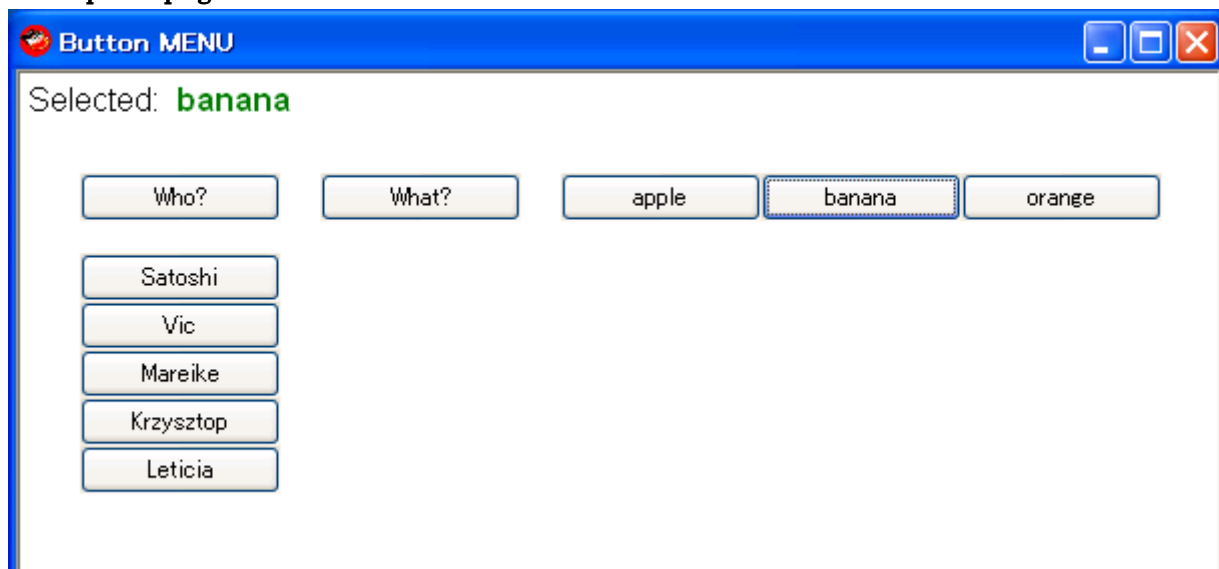
  para 'Selected: '
  @msg = para '', :stroke => green

  @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
    tb = image(:left => 0, :top => 0, :width => 100,
              :height => 21){rect(0, 0, 100, 21)}

    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,
          :fill => khaki,:curve => 8)}.hide

        yield b, f, i, e
        b.hover{f.show}
        b.leave{f.hide}
        b.click{@msg[n].text = strong(e)}
      end
    end.hide

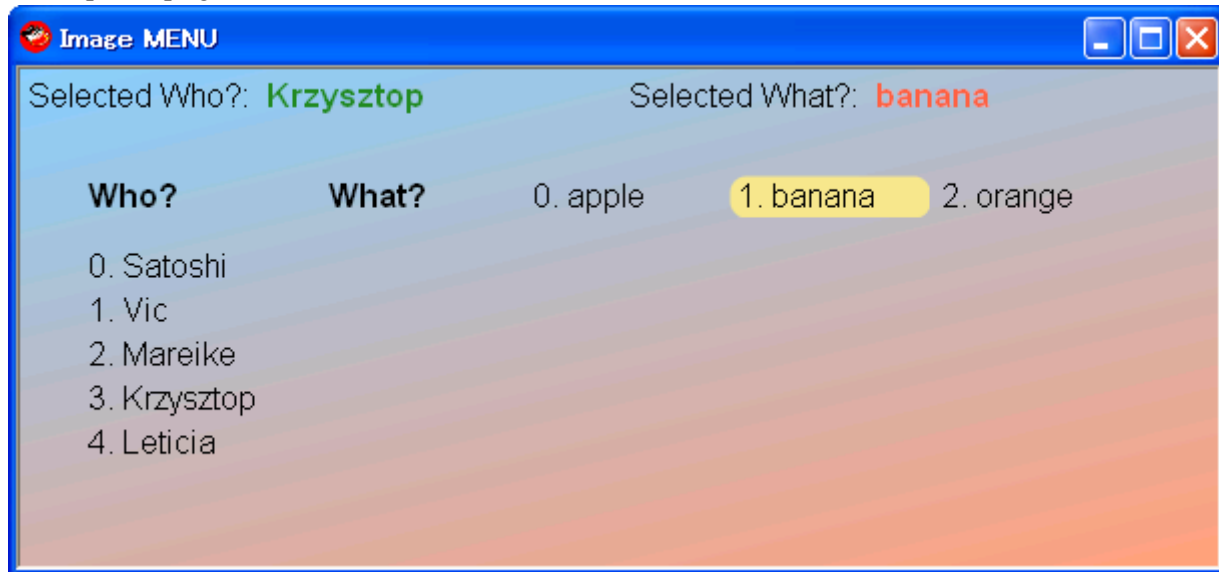
    tb.click{@f[n].toggle; @msg[n].text = ''}
  end

  @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, i, e|
      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 the items when mouse clicks on it, do the following revising and try to run.

Line No.18

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

Line No.22

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

5.19 arc and cap

New arc and cap methods are released in the 970th build.

See the following [article](http://newwws.shoooes.net/2008/09/10/arcs.html):

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

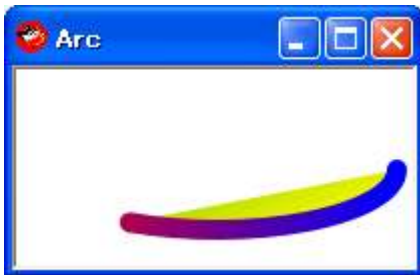
And _why shows us a wonderful combination of the animate method.

#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

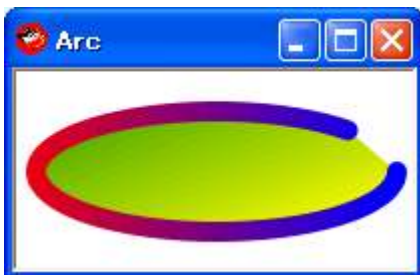
  a = animate 12 do |i|
    @c.remove if @c
    r = i * (PI * 0.01)
    @c = arc 100, 50, 180, 60, 0, i * (PI * 0.01)
    a.remove if r >= TWO_PI
  end
end
```

sample35.png



Started....

sample35-1.png



Almost finished....

5.20 widget with block

It's good using the widget object with block in the case of getting the keypress or the mouse event smoothly.

```
#sample36.rb
class Creature < widget
  def initialize
    msg = para '', :stroke => white
    c = image Dir.pwd + '/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



6. Assignment

6.1 Assignment 1 – twitter client (reader)

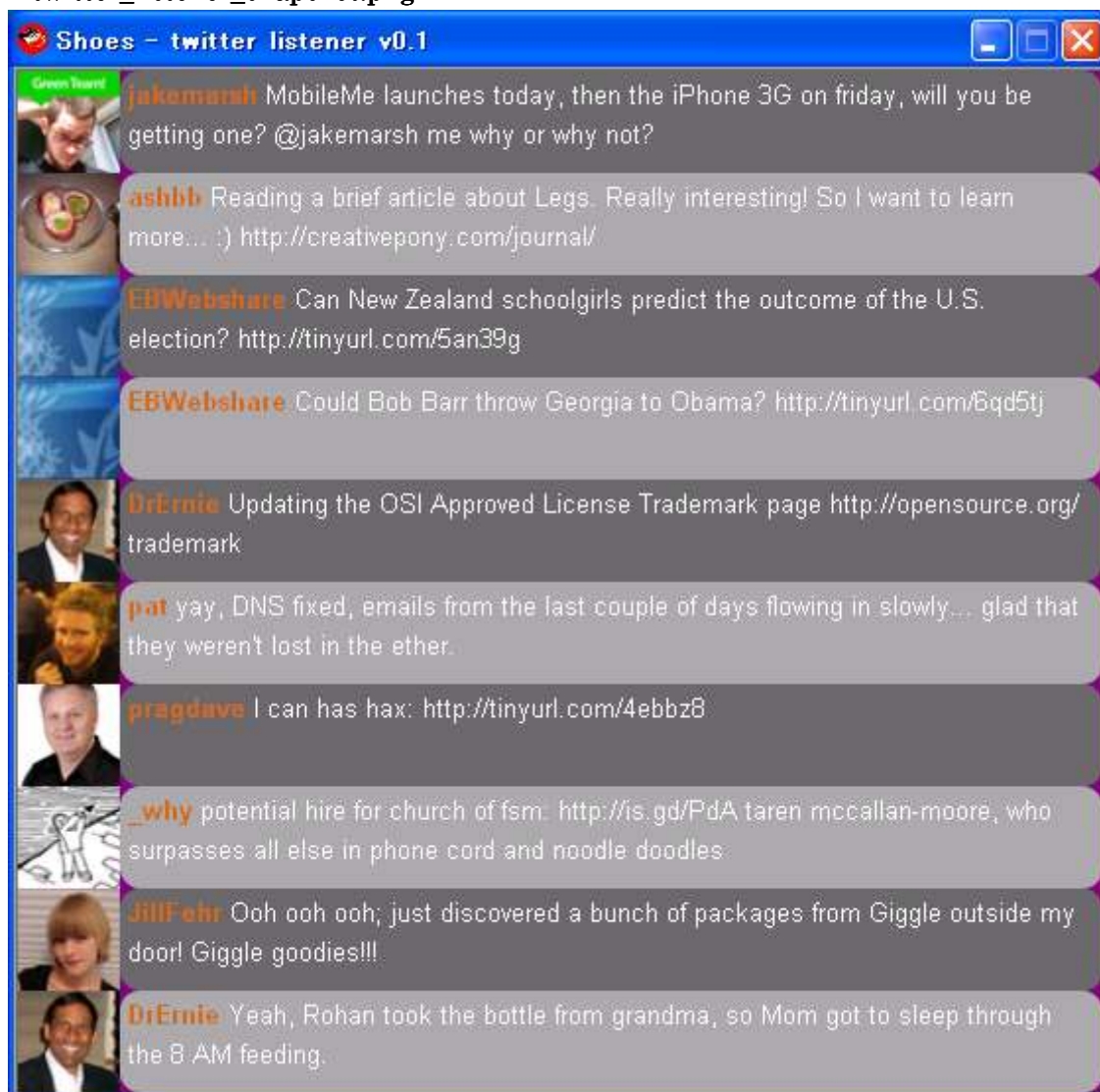
The following spec is an example.

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!

6.2 Assignment 2 – footracer

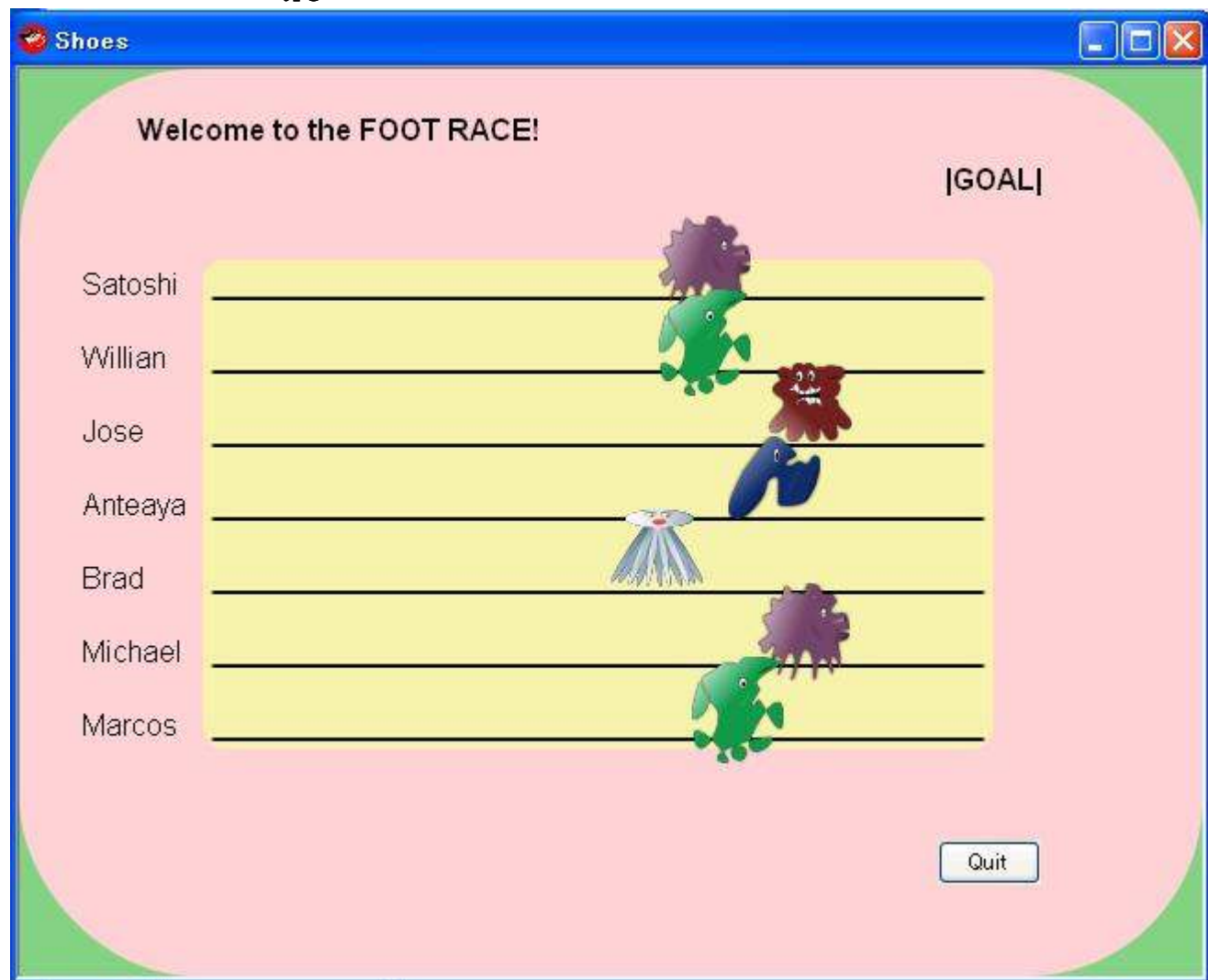
The following spec is an example.

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.jpg



Have fun!

7. Relevant web sites (Links)

- **Three manuals:** Nobody Knows Shoes (NKS) and Built-in Manual and Online Reference Manual.
<http://shoooes.net/manuals/>
<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)
(<http://www.rubyinside.com/whys-shoes-grows-up-1014.html>)

8. Appendix

8.1 Advanced article

- [Threaded XMLHttpRequest In Shoes](http://hackety.org/2008/08/15/threadedDownloadsInShoes.html)
(<http://hackety.org/2008/08/15/threadedDownloadsInShoes.html>)
- [Stamping EXEs And DMGs](http://hackety.org/2008/06/19/stampingExesAndDmgs.html)
(<http://hackety.org/2008/06/19/stampingExesAndDmgs.html>)
- [Martin DeMello's Gooley Challenge](http://hackety.org/2008/06/12/martinDemellosGooleyChallenge.html)
(<http://hackety.org/2008/06/12/martinDemellosGooleyChallenge.html>)
- [The Image Block At The Bottom Of Shoes](http://hackety.org/2008/05/22/theImageBlockAtTheBottomOfShoes.html)
(<http://hackety.org/2008/05/22/theImageBlockAtTheBottomOfShoes.html>)

8.2 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>

8.3 Shoes mailing list in Spanish

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

8.4 Shoes IRC channel

#shoes on irc.freenode.net

9. Trivia

Shoes-0.925 needs to revise the following.

- `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.

I'm not sure this behavior is Shoes new spec or bug or miss coding...

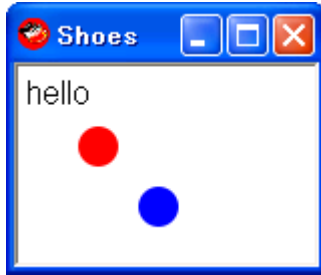
- wonder the mouse event behavior

```
# 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.

- **Shoes Fest**

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

- **Shoes was born July 31st, 2007.**

Yes, July 31st is the birthday and now one year old.

- **Shoes wiki**

[New Shoes wiki](#) was launched at Sep 12th, 2008.

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

[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 the case of Windows XP and Shoes-0.r970)

There are many sample code. Let's hack!

C:\Program Files\Common Files\Shoes\0.r970\samples

- **Lovely creatures in this tutorial created by Anita Kuno.**

Each creature has his/her own name.

- * purple is **loogink**
- * brown is **Yar**
- * green is **Cy**
- * 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 Dir.pwd +
    "/#{c}.png", :left => x += 60}

  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 Creaturs!

FIN.