GRAPHICS IN RUBY USING SHOES

Nov-2011

-

By: William Surya Permana.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 What is Shoes?

Shoes is a cross-platform toolkit for writing graphical apps easily and artfully using Ruby. Same as Ruby, Shoes is focus on simplicity and productivity. Shoes is designed to be easy and straightforward without losing power.

For more information, see http://shoesrb.com/.

1.2 Why using Shoes?

Unlike most other GUI toolkits, Shoes is designed to be easy and straightforward without losing power. Beside of that, Shoes can runs on multi-platforms like Microsoft Windows, Apple Mac OS X, Linux (GTK+), and BSD, using the underlying technologies of Cairo and Pango. With Shoes, you can also create Web-like Desktop apps.

1.3 History

Around year of 2007, why the lucky stiff, a notable person in Ruby community, created a project known as "Hackety Hack" to teach programming to everyone. In order to reach all corners of the earth, he decided to make "Hackety Hack" work on Windows, Mac OS X, and Linux. Then, _why decided to share his toolkit with the world. Thus, Shoes was born.

The first release was on July 30, 2007. Many apps were made since then, and were put into "The Shoebox". The latest version released is Shoes 3 in August 19, 2010.

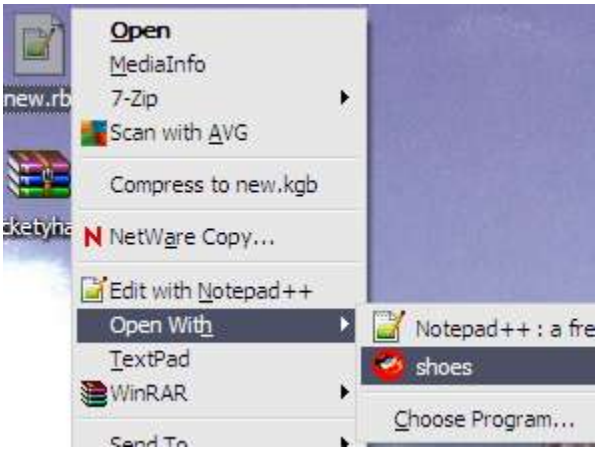
1.4 System requirement

In order to make the computer can run a Shoes program; first you must install the interpreter available here: http://shoesrb.com/downloads. Next, you can use any text editor to write the code.

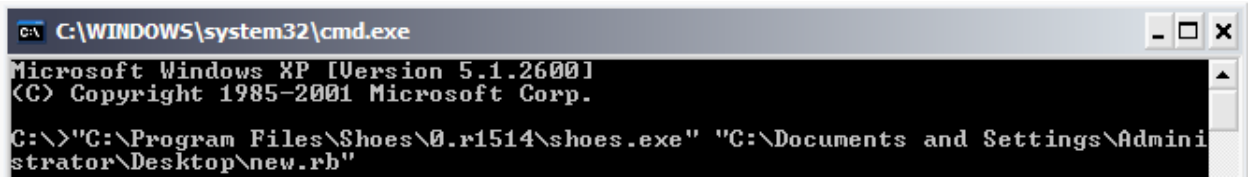
Next, to run the code, you can:

1. Open the interpreter (shoes.exe), select open an app, and browse the source file, or

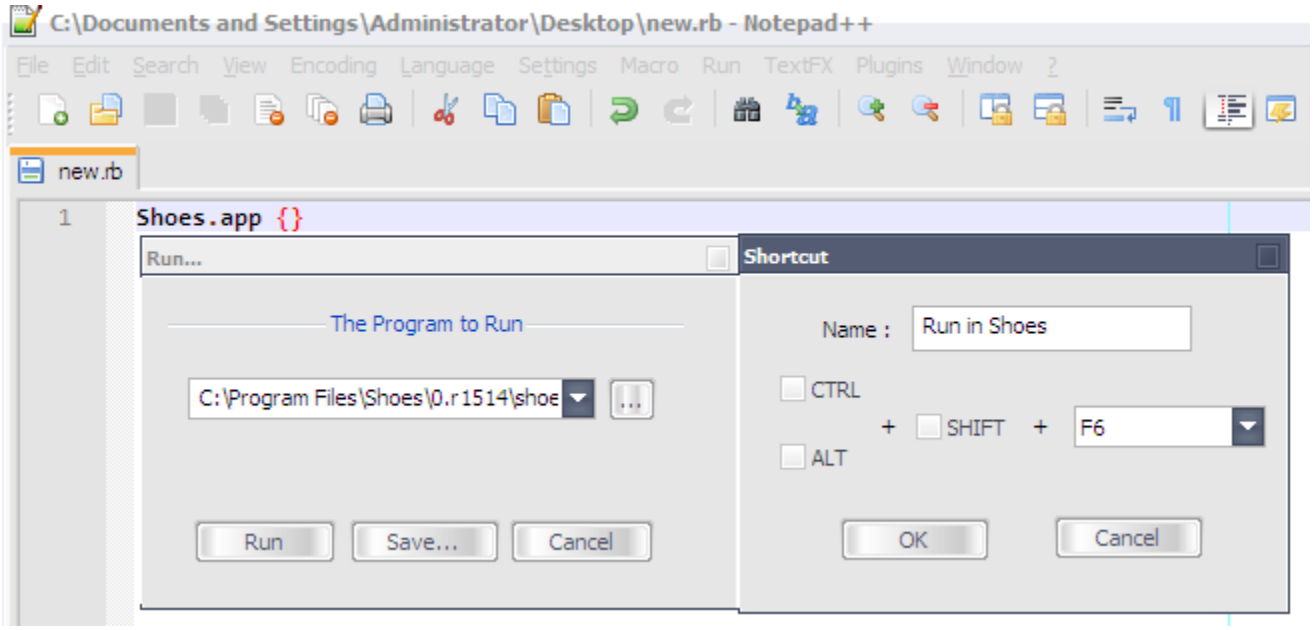
2. Open the source file with the interpreter, or



3. Using command prompt by typing shoes.exe followed by the source file, or



4. Run directly by using Notepad++, by pressing F5 and type the location of the interpreter followed by “"$(FULL_CURRENT_PATH)"”. Or, you can save this command and assign it with shortcut.



CHAPTER 2:
GETTING STARTED

2.1 Creating a new project

To get started, open a new document with Notepad++. Save as the document with .rb extension. Run it by pressing the shortcut which you have set before. You must save the rb first every time before you run it.

The code you type must be in the scope of between "Shoes.app {" or "Shop.app do" and "}" or "end".

For example,

```
Shoes.app do
    button
end
```

2.2 How to comment

In Ruby programming language you can use =begin to begin commenting some lines of syntax, and =end to end it. You can also use # in front of the text which you want to comment until the end that line.

Example:

```
=begin
 Comment line 1
 Comment line 2
=end

#Single line comment
```

CHAPTER 3:
CREATE A NEWSPAPER-LIKE DISPLAY PROGRAM

In this part of the training, you will learn how to make a GUI program that will show the news like on a newspaper, followed by a link. In this chapter, you will learn printing, styling, and lay-outing.

Look at this example:

3.1 Creating windows

Now, before we begin, let's learn about output in Shoes. To create a window, you just need to type:

```
1  Shoes.app do
2  end
```

The window can be modified by adding some properties. For example, the width, the height, the title, and whether the windows is resizable or not. For example:

```
1  Shoes.app :width=>640, :height=>640, :title=>"Latest News",
   :resizable=>false do
2  end
```

The code above will create a window with 640 pixels width and 640 pixels height with "Latest News" as its title, and also not resizable.

3.2 Shoes::TextBlock

Next, to print a text on the screen, you can use these commands:

Shoes Syntax	HTML Equivalence	Font size
banner	H1	Largest
title	H2	.
subtitle	H3	.
tagline	H4	.
caption	H5	.
para	p	Normal
inscription	H6	Small

The different is it is not written in Bold in Shoes. Besides that, you can also change the case with upcase, downcase, swapcase, and capitalize, just like in Ruby.

For example:

```
1  title "Dennis Ritchie Tutup Usia, \nPara \"Programer\" Berduka"
```

This will print on the upper left of the screen, a title which read as "Dennis Ritchie Tutup Usia, Para "Programer" Berduka".

3.3 Shoes::Text

You can also add a style to your output. For example, you want to bold some words in your outputs. For that, you can use these commands:

Shoes Syntax	HTML Equivalence	Description
strong	strong	Write the text in bold
em	em	Write the text in italic
ins	u	Write the text with underline
del	s	Write the text with strikethrough
link	a	Write the text with hover effect
code	code	Write the text in mono-space font
sub	sub	Write the text as subscript
sup	sup	Write the text as superscript

For example:

```
1  para "Area of a circle (", strong("A"), ") = ", em("π"), ".",
   em("r"), sup(" 2")
```

The code above will print "Area of a circle (**A**) = $\pi.r^2$" on the upper left of the screen.

3.4 Style

To change the style of some textblock, you can use style to define your own style. You can use these properties:

Style Properties	CSS Equivalence	Description
:weight => "bold"	font-weight:bold	Set text bold
:emphasis => "italic"	font-style:italic	Set text italic
:underline => "single"	text-decoration:underline	Write the text with underline
:font=>"[family] [style] [size]"	font:[style] [size] [family]	Write the text with strikethrough
:stroke=> [color]	color:[color]	Set text color

For example:

```
style(Tagline, :stroke=>red, :font=>"Impact italic")
```

The code above will change the style of any tagline, become red, written in italic using font Impact.

3.5 Layout

After you learned output, let's go on to how to create a layout in Shoes. There are 3 layouts in Shoes, free design, flow, and stack.

With free design you can set the position (left and top). With stack, it will add the components line per line. With flow, it will add the components horizontally. If the component hit the edge of the window, it will move down (like word wrap).

For example:

```
Shoes.app do
    stack do
        para 'A'
        para 'B'
    end

    flow do
        para 'C'
        para 'D'
   end

    para 'E', :left=>50, :top=>20
end
```

The code above will produce a window like this:

3.6 The Case

So, with that information you can make the program mentioned in the first part. If you still don't know it, here's the code:

```
1   Shoes.app :width=>640, :height=>640, :title=>"Latest News", :resizable=>false
    do
2       style(Tagline, :stroke=>red, :font=>"Impact italic")
3       style(Title, :stroke=>green)
4       style(Link, :underline=>false)
5       style(LinkHover, :underline=>false)
6
7       stack do
8           tagline "Tekno".upcase
9           title "Dennis Ritchie Tutup Usia, \nPara \"Programer\" Berduka"
10          inscription "Reza Wahyudi | Jumat, 14 Oktober 2011 | 10:47 WIB "
11          caption "KOMPAS.com"
12      end
13
14      para (
15          link("Baca selengkapnya...")
16      ), :left=>200, :top=>200
17
18      flow do
19          stack :width=>0.6 do
20              para "Setelah kepergian ", strong("Steve Jobs"), " dunia teknologi
    informasi kembali kehilangan seorang tokoh pentingnya. Dia adalah Dennis
    Ritchie, seorang ahli komputer asal Amerika, yang meninggal dunia pada Rabu
    (12/10/2011) pada usia 70 tahun."
21              para "Mungkin tak banyak yang mengenal nama Dennis Ritchie. Dia
    memang tak setenar Steve Jobs. Namun, jasanya dalam sejarah komputer dunia tak
    kalah dari Steve Jobs. Ritchie adalah orang yang membuat bahasa pemrograman C."
22          end
23          stack :width=>-0.6 do
24              image '1043402p.jpg'
25          end
26      end
27
28      flow do
29          stack :width=>50 do
30              #
31          end
32          stack :width=>-50 do
33              para "Bahasa C yang dibuat Ritchie ini menjadi dasar sistem operasi
    modern saat ini, seperti Linux, Apple MacOs, dan Android. Tanpa Ritchie,
    mungkin saja tidak ada komputer dan aplikasi-aplikasi seperti yang Anda pakai
    saat ini."
34              para "Seperti yang dikutip ", ins("Kompas.com"), " dari CNet, kabar
    kematian Ritchie pertama kali didapat dari rekan lama Ritchie, Rob Pikedi,
    melalui sebuah ", em("posting"), " di Google+. Kabar ini kemudian dikonfirmasi
    oleh pihak Bell Labs, tempat pertama di mana Ritchie memulai kariernya."
35          end
36      end
37  end
```

In this code, we create a 640x640 window which in-resizable with “Latest News” as its title. In line 2-5, we set the style of tagline: red, italic, in Impact font; title: green; and set the link with no underline.

Next, in line 7-12, it will create a stack with contains all of the titles. In 14-15, it will add a link text with absolute position: 200,200. After that it will add a flow, so anything inside it will be horizontally added. Inside that flow (line 19-25), there are two stacks, the left have 60% width, and the right is the remaining space after used by 60%. The left stack contains two paragraphs, and the right one contains an image.

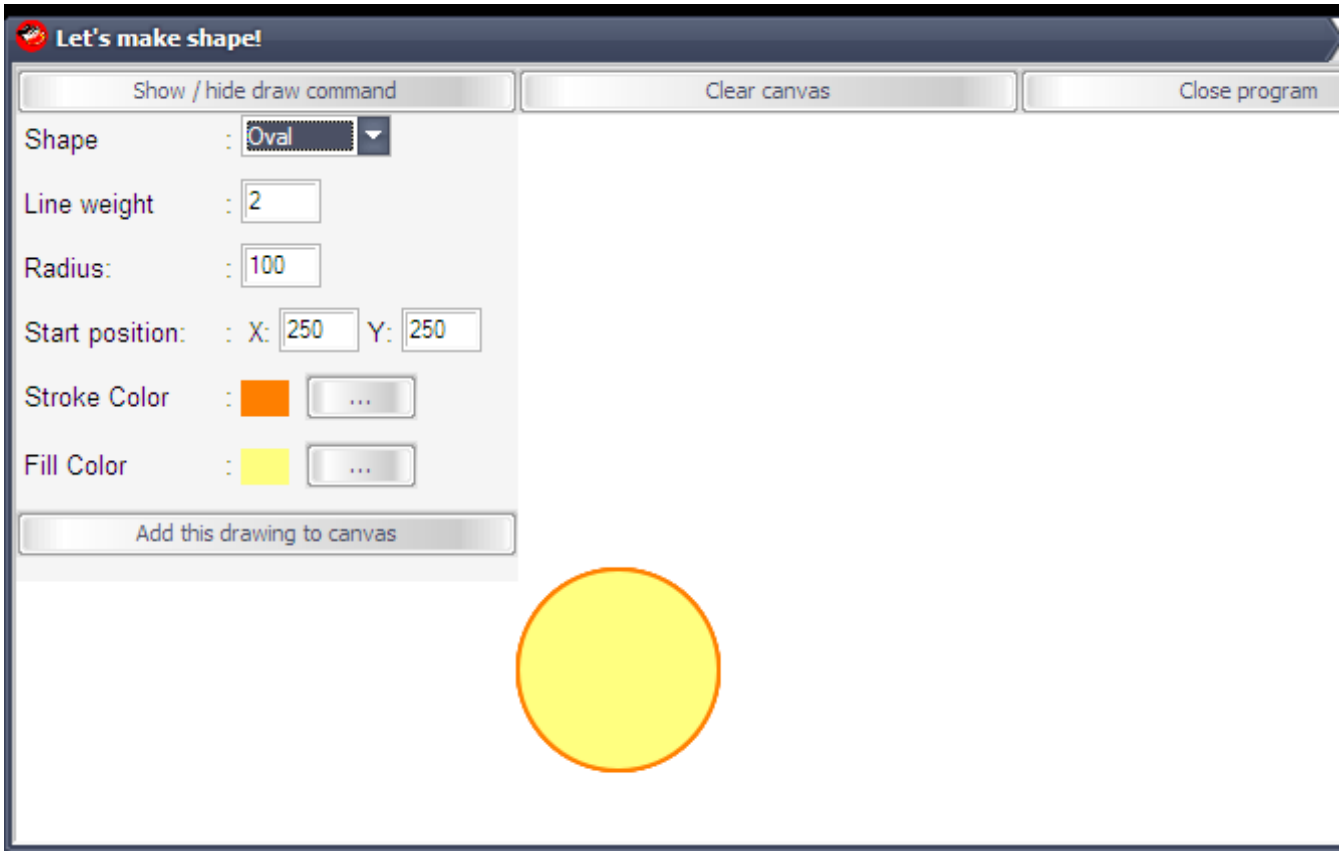
After that, it will add another flow, so anything inside it will be horizontally added too. Inside that flow (line 29-35), there are two stacks, the left have 50 pixels width, and the right is the remaining space after used by 50 pixels. The left stack is empty, but the right one contains two paragraphs.

This is the end of chapter three.

CHAPTER 4:
CREATE A SHAPE MAKER PROGRAM

In this part of the training, you will learn how to make a GUI program that includes 2 dimensional shapes and standard GUI component.

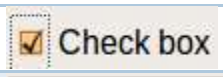
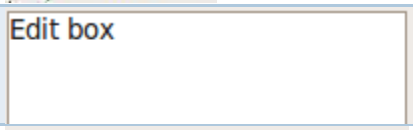
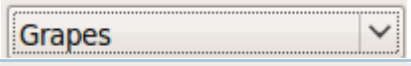
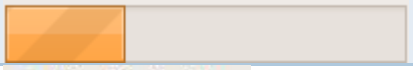
Look at this example:



4.1 Creating components

Now, before we begin, let's learn about how to draw GUI (forms) component using Shoes. In Shoes, you can draw these following components:

Component name	Sample code	Result
button	`button 'Button'`	Button

check	`check; inscription 'Check box'`	
edit_box	`edit_box 'Edit box'`	
edit_line	`edit_line 'Edit line'`	
image	`image 'shoes.png'`	
list_box	`list_box:items=>['Grapes','Apples']`	
progress	`progress`	
radio	`radio; inscription 'Radio button'`	

The code above is will create a standard component, and of course, just like TextBlock, you can set its width, height, x and y. Besides that, you can also add some statement (codes) when an action performed on that component by using do and end scope.

For example:

```
1  button 'Exit' do
2      exit
3  end
```

The code above will make a button which will close the program when clicked.

Components don't need to be named. But sometimes we must set a variable (name) to that component, so we can access that component later.

For example:

```
1  button 'Fill the progress bar!' do
2      @p.fraction = 1.0
3      alert ("The progress bar value is now "<<@p.fraction.to_s)
4  end
5  @p = progress
```

The code above will create a progress bar named @p and a button with 'Fill the progress bar!' as its text and will fill the progress bar and alert the value.

4.2 Drawing graphics

With Shoes, you can also draw primitive two dimensional shapes to the screen using these following syntaxes:

Shape	Parameters	Sample code	Result
Line	X1, y1, x2, y2	`line 0, 0, 100, 100`	
Rectangle	X1, y1, x2, y2	`rect 0, 0, 100, 100`	
Oval	X1, y1, radius	`oval 0, 0, 100`	
Arc	X1, y1, x2, y2, deg1, deg2	`arc 50, 50, 100, 100, 0, 3.14`	

4.3 The Case

So, with that information you can make the program mentioned in the first part. If you still don't know it, here's the code:

```
1   Shoes.app :width=>640, :height=>480, :title=>'Let\'s make shape!' do
2
3     def draws
4       flow do
5         button 'Show / hide draw command', :width=>250 do
6           @menu.toggle
7         end
8         button 'Clear canvas', :width=>250 do
9           clear
10          draws
11        end
12        button 'Close program', :width=>-500 do
13          exit
14        end
15      end
16      @menu = flow :width=>250 do
17        background whitesmoke
18        flow do
19          flow :width=>100 do
20            inscription 'Shape', :width=>100
21          end
22          flow :width=>-100 do
23            inscription ': '
```

```
            @shape = list_box :items => ['Line','Rectangle','Oval'],:width=>75, :choose
24  => 'Line' do
25              if @shape.text == 'Oval'
26                @divradius.show
27                @divend.hide
28              else
29                @divradius.hide
30                @divend.show
31              end
32
33            end
34          end
35        end
36        flow do
37          flow :width=>100 do
38            inscription 'Line weight', :width=>100
39          end
40          flow :width=>-100 do
41            inscription ': '
42            @weight = edit_line '1', :width=>40
43          end
44        end
45        @divradius = flow :hidden=>true do
46          flow :width=>100 do
47            inscription 'Radius:', :width=>100
48          end
49          flow :width=>-100 do
50            inscription ': '
51            @rad = edit_line '100', :width=>40
52          end
53        end
54        flow do
55          flow :width=>100 do
56            inscription 'Start position:', :width=>100
57          end
58          flow :width=>-100 do
59            inscription ': '
60            inscription 'X: '
61            @x1 = edit_line '250', :width=>40
62            inscription 'Y: '
63            @y1 = edit_line '250', :width=>40
64          end
65        end
66        @divend = flow do
67          flow :width=>100 do
68            inscription 'End position:', :width=>100
69          end
70          flow :width=>-100 do
71            inscription ': '
72            inscription 'X: '
73            @x2 = edit_line '350', :width=>40
74            inscription 'Y: '
75            @y2 = edit_line '350', :width=>40
76          end
77        end
```

```
78          flow do
79            flow :width=>100 do
80              inscription 'Stroke Color', :width=>100
81            end
82            flow :width=>-100 do
83              inscription ':'
84              @stroke = para '    ', :fill=>black
85              inscription ' '
86              button '...' do
87                @stroke.style :fill=>ask_color('Color')
88                @stroke.style :fill=>black if @stroke.style[:fill] == nil
89              end
90            end
91          end
92          flow do
93            flow :width=>100 do
94              inscription 'Fill Color', :width=>100
95            end
96            flow :width=>-100 do
97              inscription ':'
98              @fill = para '    ', :fill=>gray
99              inscription ' '
100             button '...' do
101               @fill.style :fill=>ask_color('Color')
102               @fill.style :fill=>black if @fill.style[:fill] == nil
103             end
104           end
105         end
106         flow do
107           button 'Add this drawing to canvas', :width=>250 do
108             strokewidth @weight.text.to_i
109             stroke @stroke.style[:fill]
110             fill @fill.style[:fill]
111             if @shape.text == 'Line'
112               line @x1.text.to_i, @y1.text.to_i, @x2.text.to_i, @y2.text.to_i
113             elsif @shape.text == 'Rectangle'
114               rect @x1.text.to_i, @y1.text.to_i, @x2.text.to_i, @y2.text.to_i
115             elsif @shape.text == 'Oval'
116               oval @x1.text.to_i, @y1.text.to_i, @rad.text.to_i
117             end
118           end
119         end
120       end
121     end
122
123     draws
124
125   end
```

In this code, we create a 640x480 window which in-resizable with “Let’s Make Shape!” as its title. Next, we create a function named draws, which will fill the windows. The program will call this function when start, and later in the program.

In draws function, line 4-15, we create a flow which contains three buttons. First button will toggle the visibility of menu flow. The second will clear the screen and redo the draws function. The last button will close the program.

Line 16-120 we create an other flow named menu with whitesmoke color as its background. Inside the flow we create a combo box named shape. If the text of the shape now is Oval then show flow divradius and hide flow divend. If not, then hide flow divradius and show flow divend. Divradius contains textblock and edit line for radius input. Divend contains textblock and two edit lines for ending x and y input. Beside that, we also create edit lines for line weight and start position.

We also create a color selection in line 79-104. We use a para to show the color. When user click button …, the program will ask for color and set it as para fill (background) color.

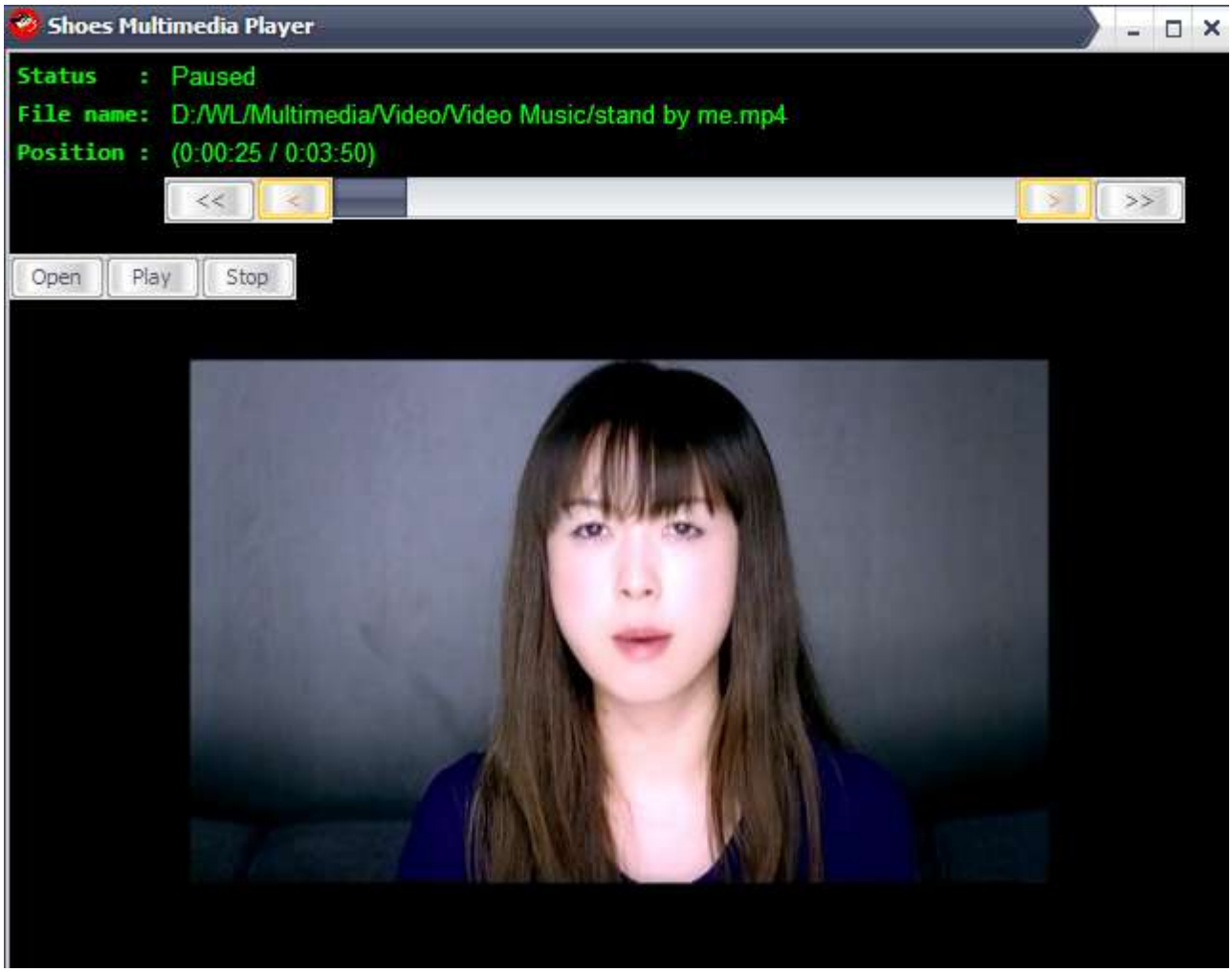
Next, we also add a button that will set the weight, color, and draw the shape.

This is the end of chapter four.

CHAPTER 5:
CREATE A MULTIMEDIA PLAYER PROGRAM

In this last chapter of the training, you will learn how to make a multimedia player program that can open, play, pause, stop, and set position of video. In this chapter, you will learn all about video element and browse a file.

Look at this example:



5.1 Browse for file

Now, before we begin, let's learn about how to browse a file, and get the file selected by user. Same as color dialog in the previous chapter, we can browse a file by using: ask_open_file. For example:

```
@fileinput = ask_open_file
```

The code above will open an open file dialog, and if user selects the file, it will be saved as fileinput variable. But will return nil if user press cancel.

5.2 Open a multimedia file.

Shoes can open common audio and video file by using video element. For example:

```
video 'stand by me.mp4'
```

The code above will draw a video from "stand by me.mp4" in the screen.

You can also use these methods to control your video:

Methods	Parameters	Function
Length		Get the total duration of the video
Pause		Pause the video
Playing?		Get the status of the video
Play		Play the video
Position		Get the video position in percent
Position	Decimal	Set the video position in percent
Stop		Stop the video
Time		Get the video position in milliseconds
Time	Number	Set the video position in milliseconds

For example:

```
@v = video 'stand by me.mp4'
```

```
@v.stop
```

The code above will open the video and stopped it.

5.3 The Case

So, with that information you can make the program mentioned in the first part. If you still don't know it, here's the code:

```
1  Shoes.app :width => 640, :height =>480, :title=>'Shoes Multimedia Player' do
2      style(Inscription, :stroke=>lime)
3      background black
4
5      stack do
6          flow :height=>20 do
7              inscription strong('Status   : '), :font=>"Consolas"
8              @status = inscription 'No audio/video selected.'
9          end
10
11         flow :height=>20 do
12             inscription strong('File name: '), :font=>"Consolas"
13             @filename = inscription ''
14         end
15
16         flow :height=>25 do
17             inscription strong('Position : '), :font=>"Consolas"
18             @position = inscription '(0:00:00 / 0:00:00)'
19         end
20
21         flow :height=>40 do
22             inscription strong('           '), :font=>"Consolas"
23             button '<<' do
24                 @v.time = 0
25             end
26             button '<' do
27                 @v.time -= 5000
28             end
29             @p = progress :width=>-280
30             button '>' do
31                 @v.time += 5000
32             end
33             button '>>' do
34                 @v.time = @v.length
35             end
36         end
37     end
38
```

```
39    stack do
40      every 1 do
41          @posnowh = (@v.time/3600000).to_s
42          @posnowm = (@v.time/60000%60).to_s
43          @posnows = (@v.time/1000%60).to_s
44          #@posnowmm = (@v.time%1000).to_s
45
46          @posnowm = "0"+@posnowm if @posnowm.length < 2
47          @posnows = "0"+@posnows if @posnows.length < 2
48          #@posnowmm = "0"+@posnowmm if @posnowmm.length == 2
49          #@posnowmm = "00"+@posnowmm if @posnowmm.length == 1
50
51          @posnow = @posnowh+':'+@posnowm+':'+@posnows
52
53          @maxpos = @v.length.to_s
54
55          @maxposh = (@v.length/3600000).to_s
56          @maxposm = (@v.length/60000%60).to_s
57          @maxposs = (@v.length/1000%60).to_s
58          #@maxposmm = (@v.length%1000).to_s
59
60          @maxposm = "0"+@maxposm if @maxposm.length < 2
61          @maxposs = "0"+@maxposs if @maxposs.length < 2
62          #@maxposmm = "0"+@maxposmm if @maxposmm.length == 2
63          #@maxposmm = "00"+@maxposmm if @maxposmm.length == 1
64
65          @maxpos = @maxposh+':'+@maxposm+':'+@maxposs
66
67          @position.text = '(' + @posnow + " / " + @maxpos + ')'
68
69          @p.fraction = @v.position
70
71          if @v.playing? == true
72              @status.text = 'Playing'
73          end
74
75          if @v.playing? == false
76              @status.text = 'Paused' if @v.time < @v.length
77              @status.text = 'Stopped' if @v.time > @v.length
78          end
79      end
80    end
81
82    flow do
83      button 'Open', :width=>50 do
84          @fileinput = ask_open_file
85          if @fileinput!=nil
86              @v.stop if @filename.text!=''
87              @v.remove if @filename.text!=''
88              @filename.text = @fileinput
89              @v = video @filename.text
90              @v.style :width=>1.0
91              @v.style :height=>-135
92              @status.text = 'Stopped'
93              @bpl.style :width=>50
```

```
 94              @bpa.style :width=>0
 95              @bst.style :width=>0
 96          end
 97      end
 98
 99      @bpl = button 'Play', :width=>0 do
100          @bpl.style :width=>0
101          @bpa.style :width=>50
102          @bst.style :width=>50
103          @v.play
104      end
105
106      @bpa = button 'Pause', :width=>0 do
107          @v.pause
108          @bpl.style :width=>50
109          @bpa.style :width=>0
110          @bst.style :width=>50
111      end
112
113      @bst = button 'Stop', :width=>0 do
114          @v.stop
115          @status.text = 'Stopped'
116          @bpl.style :width=>50
117          @bpa.style :width=>0
118          @bst.style :width=>0
119      end
120    end
121
122  end
```

In this code, first we set the style of inscription to lime color, and the background to black color. Next, in line 5-37, we add the label of status, filename, and position per total duration. We also add four buttons which will set the time of the video, and also a progress bar showing the current position of that video.

Next, we create some kind of timer which run every one second, which retrieve the current position of the video and covert it to h:mm:ss:nnn format, and also the total duration. It also set the progress bar value and the video status.

In the last flow, we create button to open the video, which will set the filename chosen by user to variable @fileinput. And if it is really chosen, the program will stop and remove the previous video (if any), set the filename status label, and play that video.

We also create four buttons for other commands, like play, pause, and stop. It also set the visibility of other buttons. For example, if we hit button stop, it will hide all other buttons, except the play button.

This is the end of chapter five.

This is the end of our training; I hope it can help you to understand the basic of Shoes.

BIBLIOGRAPHY

Web

http://shoesrb.com

http://en.wikipedia.org/wiki/Shoes_(GUI_Toolkit)

http://rubylearning.com/satishtalim/tutorial.html

Book

Why the lucky stiff. 2005. *Why's (poignant) Guide to Ruby*. A-Press.

Why the lucky stiff. 2007. *Nobody Know Shoes*.