

Alpha Version

Code of Art 2

integrating ruby-processing with your
Ruby and Rails applications

github.com/jcasimir/code_of_art2



Our Goals

- * Understand what ruby-processing does
- * See how r-p can be used for graphical prototyping
- * Become familiar with distributed processing
- * Talk about next steps

The Plan

- * What is Processing?
- * Creating Content
- * Automation
- * Next Steps

Introduction to Processing

The screenshot shows the main page of the Processing.org website. At the top, there's a navigation bar with links for Cover, Exhibition, Reference, Learning, Download, Discourse, Contribute, and About. On the right side of the top bar is a search field labeled "Search processing.org:" with a magnifying glass icon. Below the navigation bar, there's a large, stylized title "Processing" in white. To the right of the title is a search bar with the placeholder text "Search processing.org:". Below the search bar is a link to "FAQ". The main content area has two columns. The left column is titled "Exhibition" and features a thumbnail image of a complex, branching structure labeled "Branching Morphogenesis" by Sabin+Jones LabStudio. The right column is titled "» Download Processing" and contains text describing Processing as an open source programming language and environment for creating images, animation, and interactions. It mentions its use by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. The text also highlights its purpose as a sketchbook and professional production tool. Below this text is another section about downloading the software for GNU/Linux, Mac OS X, and Windows, along with links to the development repository and documentation.

What is Processing?

JEREMY/ASHKENAS

Code as literature. Give it a read.

Choose a Project:

Ruby-Processing
A Face for Stephen Hawking
Context-Free

Ruby-Processing lets you ditch Processing's crusty ol' Java syntax for a new pair of Ruby slippers. [Processing](#) is a framework for doing drawings, animations, and beautiful interactions with code — and now you can code them in Ruby. The project can be used to make web applets and cross-platform applications, and is fully open source. If you'd like to download it and get started, or even contribute, then I'll kindly direct you to the [GitHub repository](#). For comprehensive instructions and aid, check out the [wiki](#).



Ruby-Processing

Art in 2D



pointilism

Art in 3D



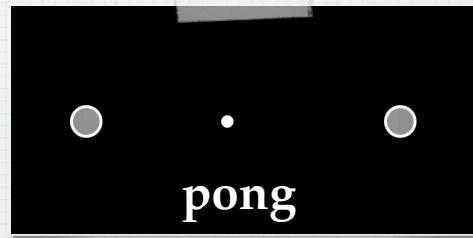
flight-patterns

Audio/Video



WPP

Interaction



pong

The Good...

...and the

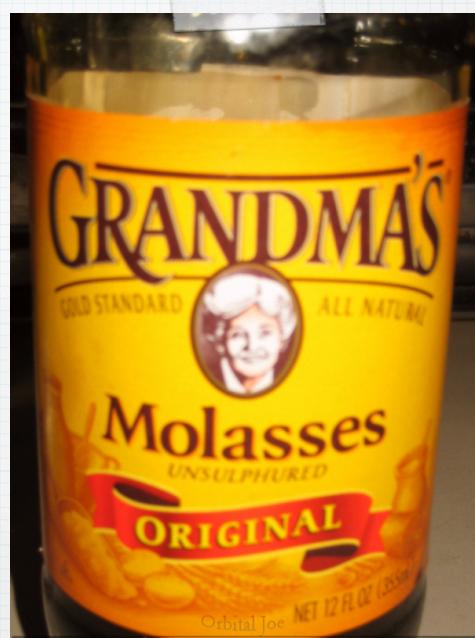
Not So Good





Ease of Development

Performance

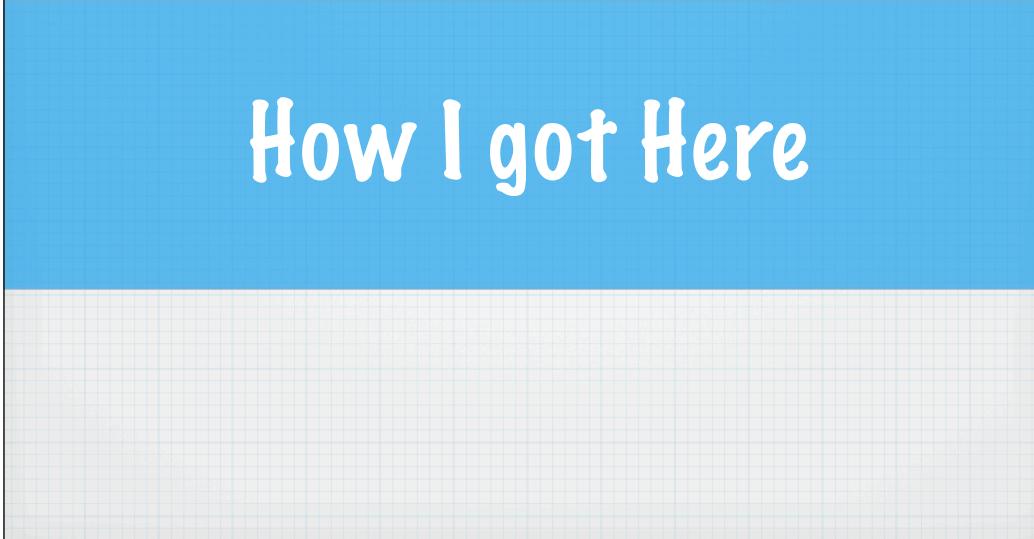




Extremely Large Data

And where does that
get you?





How I got Here

What does this icon make you *think*?



Slowdowns
Ahead!

Annoying
Interface

Who uses
desktop apps?

Excel Feature Envy

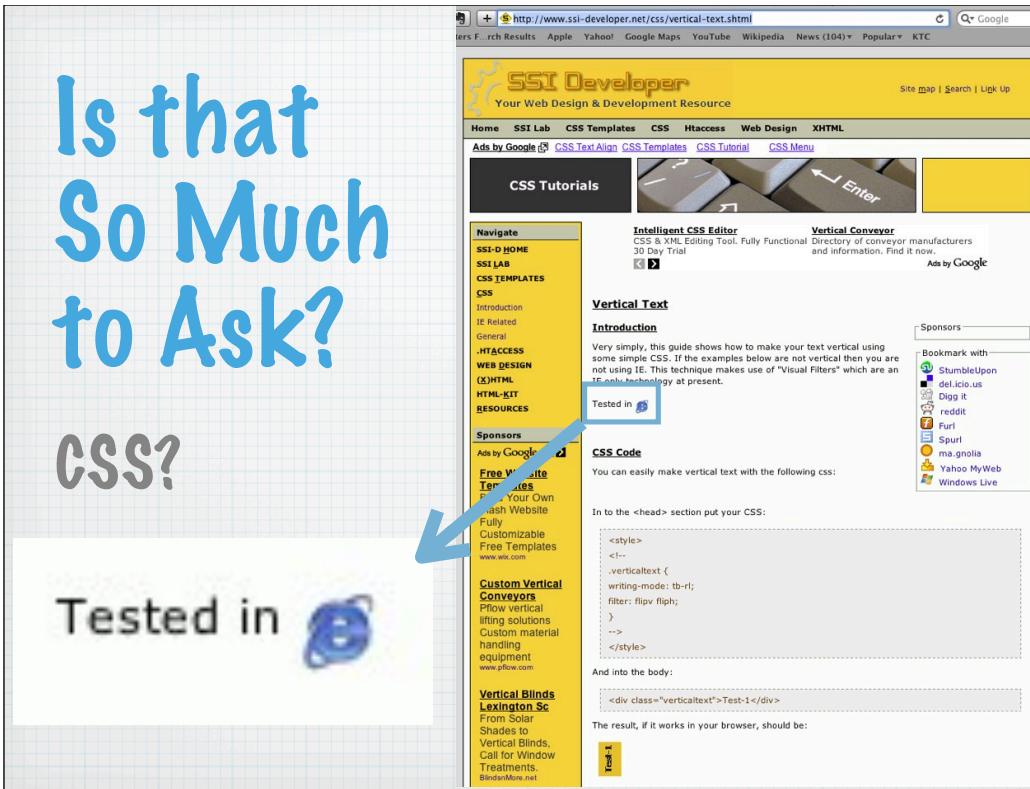
	A	B	C	D	E
1	Last Name	First Name	Week 6 Quiz	Week 7 Review Sheet	Week 8 Quiz
2	Dane	John	5	4	5
3	Goins	Sarah	4	4	5
4	Lancaster	Robert	4	2	3
5	Osprey	Karen	3	4	3
6	Samuels	Luke	2	3	4

Vertical Table Headers

Is that So Much to Ask?

css?

Tested in 



The screenshot shows a web browser window displaying the SSI Developer website at <http://www.ssi-developer.net/css/vertical-text.shtml>. The page title is "Vertical Text". A large blue arrow points from the "Tested in" text above down to the "Tested in" button on the website's sidebar.

Sidebar (Left):

- Home
- SSI Lab
- CSS Templates
- CSS
- Introduction
- IE Related
- General
- .HTACCESS
- WEB DESIGN
- (S)HTML
- HTML-KIT
- RESOURCES

Sponsors:

- Free Website Templates
- Create Your Own Flash Website
- Fully Customizable
- Free Templates
- www.wix.com

Custom Vertical Conveyor:

- Offering vertical lifting solutions
- Custom material handling equipment
- www.pilow.com

Vertical Blinds Lexington Sc:

- From Solar Shades to Vertical Blinds, Call for Window Treatments.
- [BlindeMore.net](http://BlindenMore.net)

Tested in

Content Area (Right):

Intelligent CSS Editor
CSS & XML Editing Tool. Fully Functional 30 Day Trial

Vertical Conveyor
Directory of conveyor manufacturers and information. Find it now!

Vertical Text

Introduction

Very simply, this guide shows how to make your text vertical using some simple CSS. If the examples below are not vertical then you are not using IE. This technique makes use of "Visual Filters" which are an IE-only technology at present.

Tested in

CSS Code

You can easily make vertical text with the following css:

```
<style>
<!--
.verticaltext {
writing-mode: tb-rl;
filter: flipv fliph;
}
-->
</style>
```

In to the <head> section put your CSS:

```
<div class="verticaltext">Test-1</div>
```

And into the body:

```
<div class="verticaltext">Test-1</div>
```

The result, if it works in your browser, should be:

Test-1

Is that So Much to Ask?

CSS Transforms?

top left	style on cell		style on div	style on span
centre				
bottom				

Is that
So Much
to Ask?

CSS

Javascript

SVG

Canvas

Conclusion:

I need to dynamically
generate images
based on data
from my Rails application.

Creating Images with Ruby-Processing

Step 1: Proof of Concept

* rp_sample.rb

```
1 def setup
2   @x_max = 100
3   @y_max = 200
4   size @x_max, @y_max
5 end
6
7 def draw
8   background 255
9   fill rand(255),rand(255),rand(255)
10  oval 50,100,50, 50
11  save "sample.png"
12  exit
13 end
```

rp5 rp_sample.rb

Step 2: Goal Setting

Provided: "Week 6 Quiz"

Create:

WEEK 6 QUIZ

- Vertical Text
- Sized by Text Length
- Use Silkscreen Font
- Defaults:
 - Size 8px
 - Black text on white

Step 3: R-P Algorithm

1. Build superstructure
2. Set options
3. Create the window
4. Move the origin to the bottom
5. Rotate the plane
6. Write the text
7. Save the file

Step 4: Implementation

1. Build Superstructure

```
* rp_text_generator_step1.rb
1 def setup
2
3 end
4
5 def load_parameters
6
7 end
8
9 def draw
10
11 end
```

2. Set Options & 3. Create Window

```
* rp_text_generator_step2.rb
1 def setup
2   @params = load_parameters
3   size @params[:width], @params[:height]
4 end
5
6 def load_parameters
7   defaults = {
8     :width => 100,
9     :height => 100,
10    :background => 255
11  }
12 end
13
14 def draw
15   background @params[:background]
16 end
```

4. Move the Origin

```
def load_parameters
  defaults = {
    :width => 100,
    :height => 100,
    :background => 255,
    :line_height => 8
  }
end

def draw
  background @params[:background]

  translate @params[:line_height] -2, @params[:height]
end
```

Origin
Moves

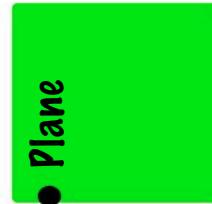
New
Code

5. Rotate the Plane

```
def load_parameters
  defaults = {
    :width => 100,
    :height => 100,
    :background => 255,
    :line_height => 8,
    :rotation = radians -90
  }
end

def draw
  background @params[:background]

  translate @params[:line_height] -2, @params[:height]
  rotate @params[:rotation]
end
```



New
Code



6. Writing Text

```
:rotation = radians -90,  
:font = load_font "fonts/Silkscreen-8.vlw",  
:font_color = 0  
:text = "default text"  
}  
end  
  
def draw  
background @params[:background]  
  
translate @params[:line_height] -2, @params[:height]  
rotate @params[:rotation]  
  
fill @params[:font_color]  
text_font @params[:font]  
text @params[:text], 0, 0  
end
```

Processing
Font File

New Code

7. Save the File

```
:text => "default text",
:file_name => "default",
:file_type => "png"
}
end

def draw
background @params[:background]

translate @params[:line_height] -2, @params[:height]
rotate @params[:rotation]

fill @params[:font_color]
text_font @params[:font]
text @params[:text], 0, 0

save @params[:file_name] + "." + @params[:file_type]
exit
end
```

New
Code

Step 5: Refactor

1. Condense Window Size

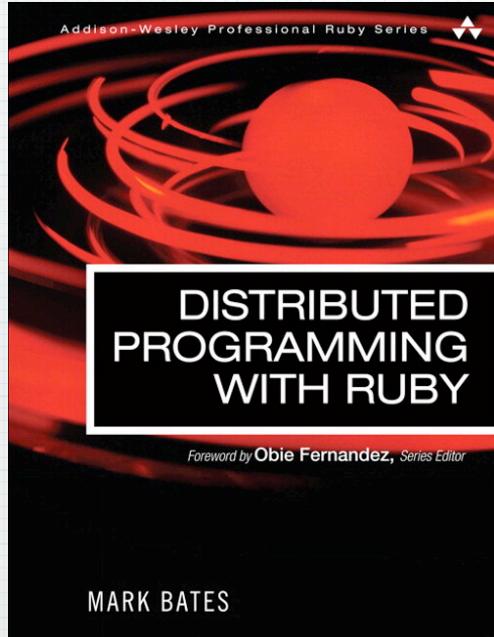
```
def load_parameters
  defaults = {
    :background => 255,
    :line_height => 8,
    :character_width => 5, Additional Parameter
    :rotation => radians(-90),
    :font => load_font("fonts/Silkscreen-8.vlw"),
    :font_color => 0,
    :text => "default text",
    :file_name => "default",
    :file_type => "png"
  }

  defaults[:width] = defaults[:line_height]
  defaults[:height] = (defaults[:text].length + 1) * defaults[:character_width]
  defaults
end
```

Calculated to fit

Parameters from Our Rails App

How can we pass
messages from Rails
to Ruby-Processing?



- Mark Bates
- Published 11/09
- Best tutorial for dRB, Rinda, etc



Beanstalkd

“Beanstalk
is a simple,
fast
workqueue
service.”

Installing Beanstalkd

MacOS X

```
brew install beanstalkd  
sudo port install beanstalkd
```

Fedora Linux

```
yum install beanstalkd
```

From Source

<http://github.com/kr/beanstalkd>

Windows??

<http://github.com/ghazel/beanstalkd>

Running Beanstalkd

beanstalkd

(that's it!)

Defaults:

- Listen on 0.0.0.0:11300
- Run as current user
- Stay attached

Beanstalkd from Ruby

```
gem install beanstalk-client
```

```
jc-iMac:Casimir Creative jcasimir$ irb
>> require 'beanstalk-client'
=> true
>> bs = Beanstalk::Pool.new(['0.0.0.0:11300'])
=> #<Beanstalk::Pool:0x100541770 @default_tube=r
t"], @last_used="default", @socket=#<TCPSocket:0
>> bs.put("This is the body of my job")
=> 28
>> [
```

Adding a Job to the Queue

Pulling a Job from the Queue

```
jc-iMac:Casimir Creative jcasimir$ irb
>> require 'beanstalk-client'
=> true
>> bs = Beanstalk::Pool.new(['0.0.0.0:11300'])
=> #<Beanstalk::Pool:0x100541770 @default_tube=
t"], @last_used="default", @socket=#<TCPSocket:
>> job = bs.reserve
=> (job server=0.0.0.0:11300 id=28 size=26) - St
>> puts job.body
This is the body of my job
=> nil
>> job.delete
=> true
>> █
```

What if there's no Job?

```
>> job = bs.reserve
```



It waits!

How will this work?

Assignment Model

- When an assignment is saved...
 - Find the Beanstalk
 - Create a Job and supply the...
 - file_name
 - text

How will this work?

Ruby-Processing Worker

- Wait for a Job to hit the queue
- Load the parameters from the job body
- Create the image
- Delete the job
- Wait for the next job

Extra Challenges?

Beanstalkd works with plain
“STRINGS”

For more complex objects,
we’ll need to serialize/unserialize

YAML is easy and fast

Implementing the Beanstalk Solution

Hooking into the Assignment Model

```
class Assignment < ActiveRecord::Base
  attr_accessible :name
  has_many :grades

  after_save :generate_header_image
  def generate_header_image
    bs = Beanstalk::Pool.new(['0.0.0.0:11300'])
    body = { :file_name => image_file_name, :text => self.name }.to_yaml
    bs.put(body)
  end

  def image_file_name
    "assignment_id_#{self.id}"
  end
```

(that's the easy part)

R-P Can't Easily Use Gems

```
jc-iMac:code jcasimir$ mkdir vendor  
jc-iMac:code jcasimir$ cd vendor  
jc-iMac:vendor jcasimir$ mkdir gems  
jc-iMac:vendor jcasimir$ cd gems  
jc-iMac:gems jcasimir$ gem unpack beanstalk-client  
Unpacked gem: '/Users/jcasimir/Dropbox/Presentations/  
jc-iMac:gems jcasimir$ █ question...
```

Unpack the beanstalk-client gem into
vendor/gems/beanstalk-client-1.0.2

R-P Can't Easily Use Gems

```
1 Dir.glob(File.join("vendor", "gems", "*", "lib")).each do |lib|
2   $LOAD_PATH.unshift(File.expand_path(lib))
3 end
4
5 require 'beanstalk-client'
6 require 'yaml'
7
```

- Modify the \$LOAD_PATH to find the new vendor folder
- Require beanstalk and yaml

Modifying the setup

```
def setup
  #@params = load_parameters
  #size @params[:width], @params[:height]
  size 200, 200
  @beanstalk = Beanstalk::Pool.new(['0.0.0.0:11300'])
end
```

- Set an arbitrary window size
- Find the beanstalk queue

Modifying load_parameters

```
def load_parameters()
    defaults = {
        :background => 255,
        :line_height => 8,
        :character_width => 5,
        :rotation => radians(-90),
        :font => load_font("fonts/Silkscreen-8.vlw"),
        :font_color => 0,
        :text => "default text",
        :file_name => "default",
        :file_type => "png"
    }
```

Leave the defaults alone

Modifying load_parameters

```
job = @beanstalk.reserve  
values = defaults.merge(YAML.load(job.body))  
job.delete
```

1. Wait for a job with reserve
2. When a job is found...
 - a. Get the body
 - b. un-YAML it
 - c. merge with defaults
3. Remove the job

Modifying load_parameters

```
values[:width] = values[:line_height]
values[:height] = (values[:text].length * values[:character_width] * 1.07).to_i
values
end
```

1. Change defaults in calculations to values
- b. Changed the scaling formula slightly (not important)

Changes to draw

```
39 def draw
40     @params = load_parameters
41     size @params[:width], @params[:height]
42     background @params[:background]
43 
```

1. Load @params each time draw runs instead of once in setup
2. Set the image size after each params load

Changes to draw

```
44 |     translate @params[:line_height] - 2, @params[:height]
45 |     rotate @params[:rotation]
46 |
47 |     fill @params[:font_color]
48 |     text_font @params[:font]
49 |     text @params[:text], 0, 0
```

No change to the text rendering

Changes to draw

```
51 save TARGET_DIR + @params[:file_name] + "." + @params[:file_type]  
52 end
```

1. Add the TARGET_DIR to the save call
2. Remove the exit command so Ruby-Processing keeps looping draw

Setting File Paths

```
8 TARGET_DIR = "./tabler/public/images/generated/"
```

- Set a target directory constant just below the requires
- Make sure the directory exists

Getting Ready to Test

1. Make sure beanstalkd is running
2. Start up the tabler (Rails) server
3. Start the image processor with
`rp5 run rp_text_generator.rb`
4. Start a script/console session for experimenting

Testing from Console

1. Create a New Assignment

```
Assignment.create(:name =>  
  "My Assignment")
```

2. Look for an assignment_id_x.png in tabler/public/images/generated that has the text My Assignment

3. Update all images with:

```
Assignment.all.each{|a| a.save}
```

Implementing in the View

1. Open the view

assignments/index.html.haml

2. Change the %th that outputs the assignment name to:

```
%th= image_tag(a.image_path)
```

3. Add image_path method to Assignment

```
16 def image_path  
17   "generated/#{image_file_name}.png"  
18 end
```

Ta-Da!

Try creating new assignments and
editing existing assignments through
the web interface

What Have We Done?

- Built a distributed message passing framework
- Built a Ruby-Processing powered worker process
- Rolled it all into a Rails app

So what now?

**“I was promised
video!”**

Video in Ruby-Processing

Demo of
video_tagger.rb

Next Steps

On the Server Side

Xvfb
god

Going Beyond Prototypes

Replace the Workers,
Keep the Process

Efficient Workers

2D Images

Textorize
ImageMagick

3D Images

Ogre3D
OpenGL

Audio & Video

ffmpeg
Panda

Wrapup

The Links

Presentation & Code	http://github.com/jcasimir/Code-of-Art-2
Processing	http://processing.org
Ruby-Processing	http://github.com/jashkenas/ruby-processing
Learning Processing (book)	http://rubyurl.com/3Tu6
Learning Processing (examples)	http://github.com/jashkenas/learning-processing-with-ruby
Beanstalk	http://kr.github.com/beanstalkd/
Beanstalk-Client	http://beanstalk.rubyforge.org/
Silkscreen	http://kottke.org/plus/type/silkscreen/



JumpstartLab

 **RubyJumpstart**

Intro to Programming with Ruby
April 17 & 18

 **FlexJumpstart**

Intro to Design with Adobe Flex
April 24 & 25

 **RailsJumpstart**

Level 2 Rails Development
(BDD, ActiveRecord, HAML)
May 22 & 23



THE
HASHROCKET
WAY

Professional Rails Development
June 6 in Baltimore

Thanks!
