Ray: Ruby gaming

2D and 3D gaming in ruby

Matthew Bennett
matthew at quickwebdesign.net
@undecisive

Or...

Ray: The 80's Way

Really tacky* games development in ruby

* because of my lack of skill, no reflection on Ray...

Has the presentation started?

Probably not.

Installation

- In Linux:
  - apt-get install libsndfile1 libglew1.5-dev glew-utils libftgl2 libfreetype6 libfreetype6-dev libopenal-dev libopenal1 libsndfile1-dev
- On mac or windows:
  - Instructions on the website:
  - http://mon-ouie.github.com/projects/ray.html
- On all:
  - **gem install ray**

Getting started

- RDoc incomplete
- Good intro between website and github page

Boilerplate

```
 1 require 'rubygems'
 2 require 'ray'
 3
 4 Ray.game "My game", :size => [800, 800] do
 5   register { add_hook :quit, method(:exit!) }
 6
 7   scene :some_scene do
 8   end
 9
10   scenes << :some_scene
11 end
```

Better Boilerplate

```
 1 require 'rubygems'
 2 require 'ray'
 3
 4 class SomeScene < Ray::Scene
 5   scene_name :title
 6
 7   def setup...
 8   def register...
 9   def render(win)...
10   def clean_up...
11 end
```

… and then

```
13 class Game < Ray::Game
14   def initialize
15     super "Awesome Game"
16     SomeScene.bind(self)
17     ...
18     scenes << :title
19   end
20 end
```

Why scenes << title ?

```
1  require 'rubygems'
2  require 'ray'
3
4  class SomeScene < Ray::Scene
5    scene_name :title
6
7    def setup...
8    def register...
9    def render(win)...
10   def clean_up...
11 end
```

Event handling

- Nice easy DSL

```
on :mouse_motion do |pos|
  @rect.pos = pos
end
```

  - pos is a Vector2(D), with pos.x and pos.y
  - key_press alternative:

```
always do
  puts "holding a" if holding? :a
```

Pollygone-crackers

- Polygons are easy:

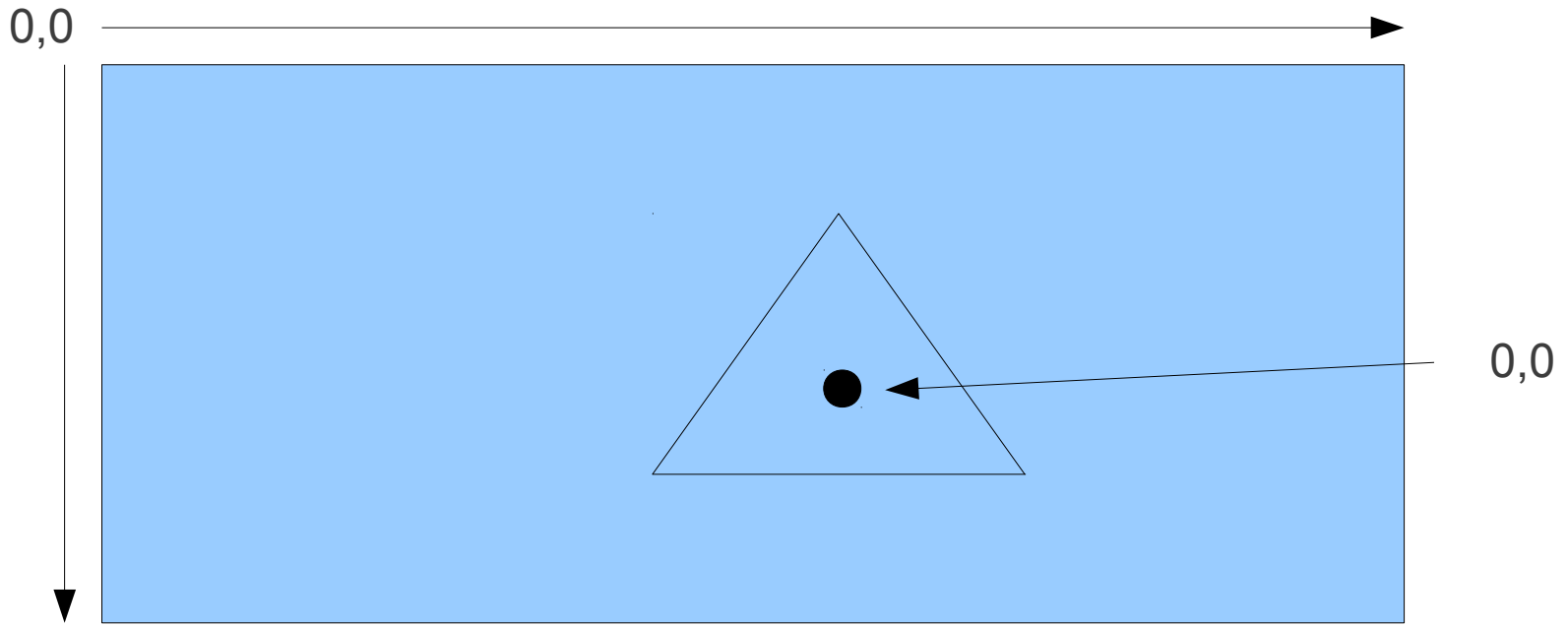
```
@rect = Ray::Polygon.rectangle(
          [-10, -10, 20, 20],
          Ray::Color.red)
```

- Rendering is easy:

```
win.draw @rect
```

Polygon Points

- Relative to the polygon position
- Retrieving is different to setting



Quick example

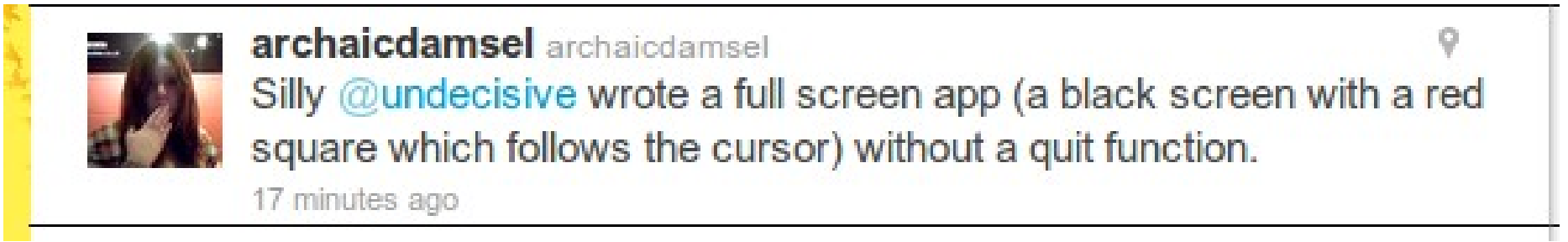
Bang, and the program's gone...

How's about some fullscreen 3D?

Best tip: Add something like this...

```
def register
  add_hook :key_press, method(:exit!)
end
```

What kind of idiot would forget a thing like that?



Fullscreen goodness

```
class Game < Ray::Game
  def initialize
    super "Awesome Game", :fullscreen => true, :size => Ray.screen_size
```

```
Ray.game "Fullscreen", :fullscreen => true, :size => Ray.screen_size do
```

Yeah, but you promised 3D...

- 3D is hard.

Things I didn't have time for

- Off-screen rendering
- Viewports
- Audio (you're probably hearing an example right now)
- Testing support
- 3D rendering
- Vertices
- Textures
- Shaders
- Et al

References

- https://github.com/Mon-Ouie/ray
- http://mon-ouie.github.com/projects/ray.html
- https://github.com/undecided/ray_examples

Shameless Plug / RFC

Talking of 80-esq graphics...

Are you locationally challenged?

Do you have a team?
Do you know people?

Are you retro, dude?

Do you do retrospectives?
Do you find it difficult with distributed teams?

Retrodude.com

Retrospectives made waaaaay too simple

Retrodude

Retrospectives made waaaaay too simple

Our survey says...

Here are the results of the retrospective so far. The list should update as the results come in - refresh the page if you're impatient!

Edit my answers

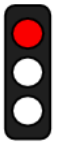
Export as tikiwiki

Phillip Jane

Do more

Do less

Start doing



Stop doing

Keep doing

Retrospectives using Retrodude	PHP	Being efficient in your retrospectives	CONFUSION!!	Rollin' rollin' rollin...
	Work related		Shaving Yaks	

Retrodude

Retrospectives made waaaaay too simple

Does it work for you?
What would you need from a tool like that?

All suggestions to @undecisive

Any questions?