

Basking in the



limelight

by Micah Martin
8th Light, Inc.



Limelight is....



(Ruby on the Desktop)

Ideas From The Web



Ideas From The Web



**THE
GOOD**

Copyright 2009 Micah Martin

Ideas From The Web



**THE
GOOD**



**THE
BAD**

Ideas From The Web



THE GOOD AND THE UGLY AND THE BAD

Copyright 2009 Micah Martin

A photograph of a theater stage. The stage is framed by heavy, red, pleated curtains that are pulled back on both sides. The floor of the stage is a light-colored wood, and a single spotlight illuminates a rectangular area in the center. The background is a dark, textured wall. The overall atmosphere is dramatic and theatrical.

Theater Metaphor



Macintosh HD

Desktop



The Making of an ~~Application~~ a Production

Setup

1. Install JRuby

2. Install Limelight Gem

- `jruby -S gem install gemcutter`
- `jruby -S gem tumble`
- `jruby -S gem install limelight`



JRuby syntax to
execute gem
executable



limelight command to
generate production
files



Name of
production to
create



```
1 $ jrubby -S limelight create production example
2   creating directory: ./example
3   creating file:      ./example/production.rb
4   creating file:      ./example/stages.rb
5   creating file:      ./example/styles.rb
6   creating directory: ./example/spec
7   creating file:      ./example/spec/spec_helper.rb
8   creating directory: ./example/default_scene
9   creating file:      ./example/default_scene/props.rb
10  creating file:      ./example/default_scene/styles.rb
11  creating directory: ./example/default_scene/players
12  creating directory: ./example/spec/default_scene
13  creating file:      ./example/spec/default_scene/default_scene_spec.rb
```



output from command

Opening a Production

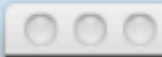
```
1 $ jruby -S limelight open example
```



limelight command to
generate production
files




Name of
production to
create



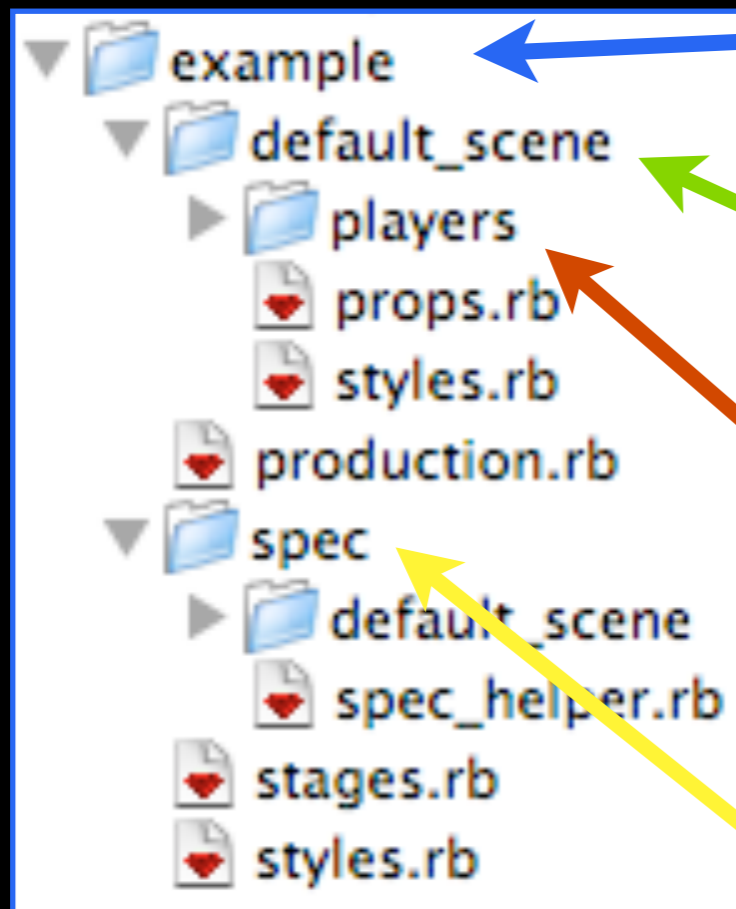
Limelight

This is the Default Scene scene.



The Structure and Interpretation of Limelight Productions

Directories



production root
(named after production)

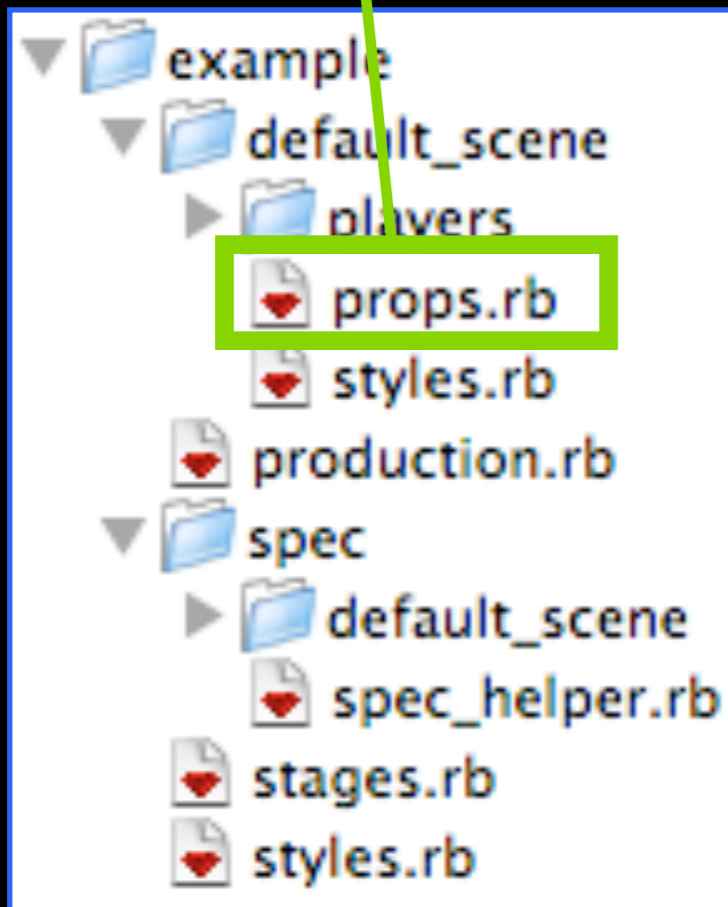
a scene
(named "default_scene")

players directory
(for the scene)

specs
(you do TDD right?)

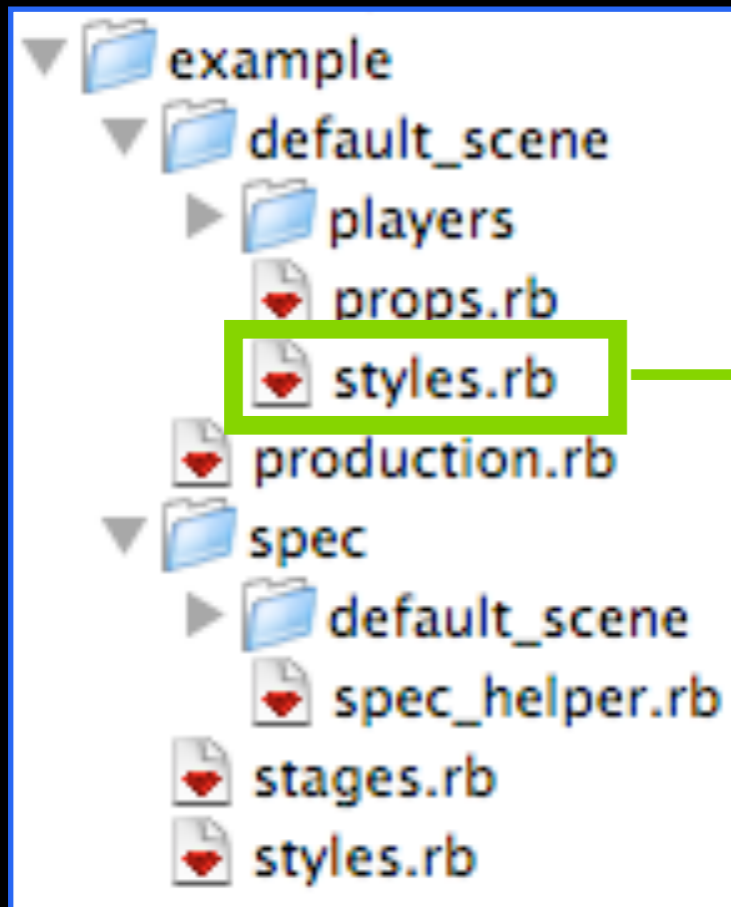
props.rb / Prop DSL

```
1 root :text => "This is the Default Scene scene."
```



- Define the Structure of a Scene
- All Ruby Code
- DSL to build tree structure of props
- Arbitrary Prop names

styles.rb / Styles DSL



```
1 default_scene {
2   background_color :black
3   horizontal_alignment :center
4   vertical_alignment :center
5   width "100%"
6   height "100%"
7 }
8
9 root {
10  text_color :white
11  font_size 18
12 }
```

- Define the Look and Feel of a Scene
- All Ruby Code
- key:value based DSL

Naming conventions

styles.rb

```
1 default_scene {
2   background_color :black
3   horizontal_alignment :center
4   vertical_alignment :center
5   width "100%"
6   height "100%"
7 }
8
9 root {
10  text_color :white
11  font_size 18
12 }
```

Naming conventions

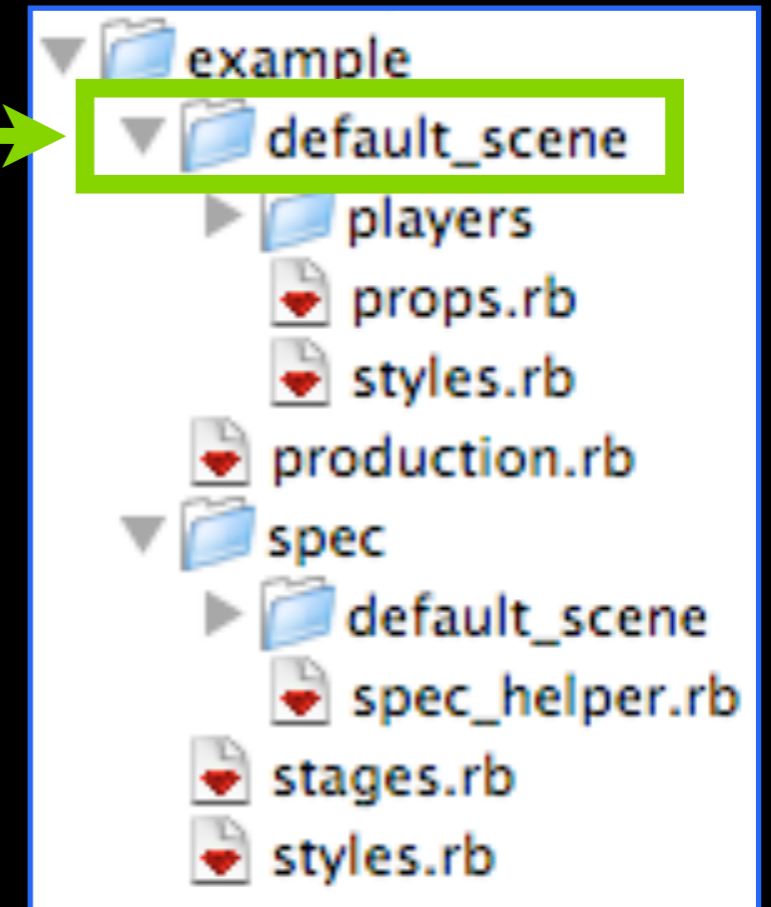
styles.rb

```
1 default_scene {
2   background_color :black
3   horizontal_alignment :center
4   vertical_alignment :center
5   width "100%"
6   height "100%"
7 }
8
9 root {
10  text_color :white
11  font_size 18
12 }
```

Naming conventions

styles.rb

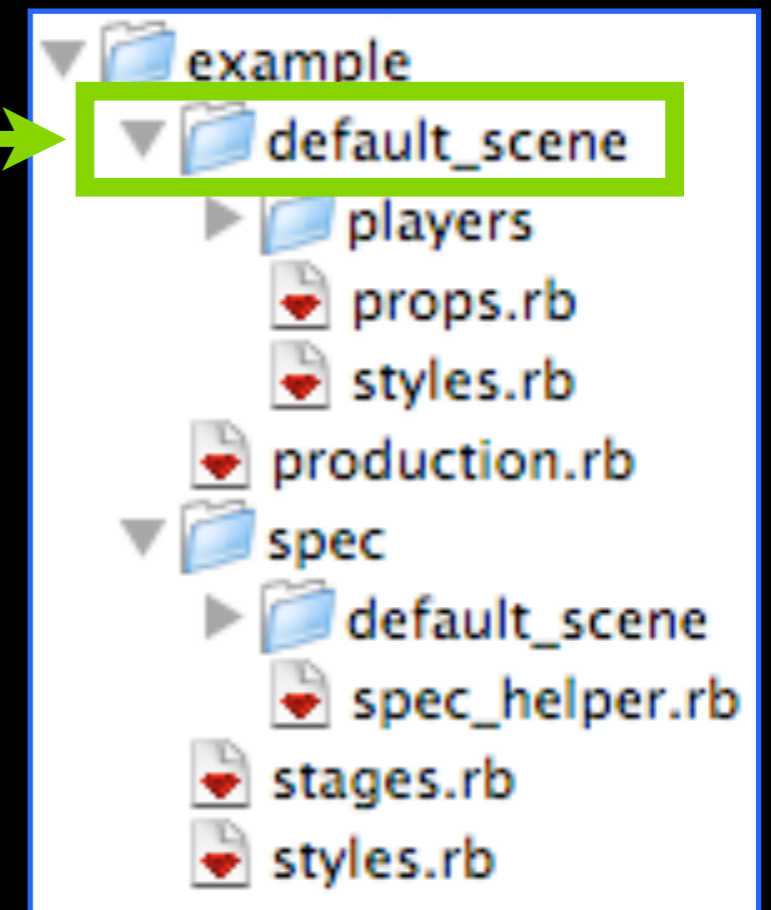
```
1 default_scene {  
2   background_color :black  
3   horizontal_alignment :center  
4   vertical_alignment :center  
5   width "100%"  
6   height "100%"  
7 }  
8  
9 root {  
10  text_color :white  
11  font_size 18  
12 }
```



Naming conventions

styles.rb

```
1 default_scene {  
2   background_color :black  
3   horizontal_alignment :center  
4   vertical_alignment :center  
5   width "100%"  
6   height "100%"  
7 }  
8  
9 root {  
10  text_color :white  
11  font_size 18  
12 }
```



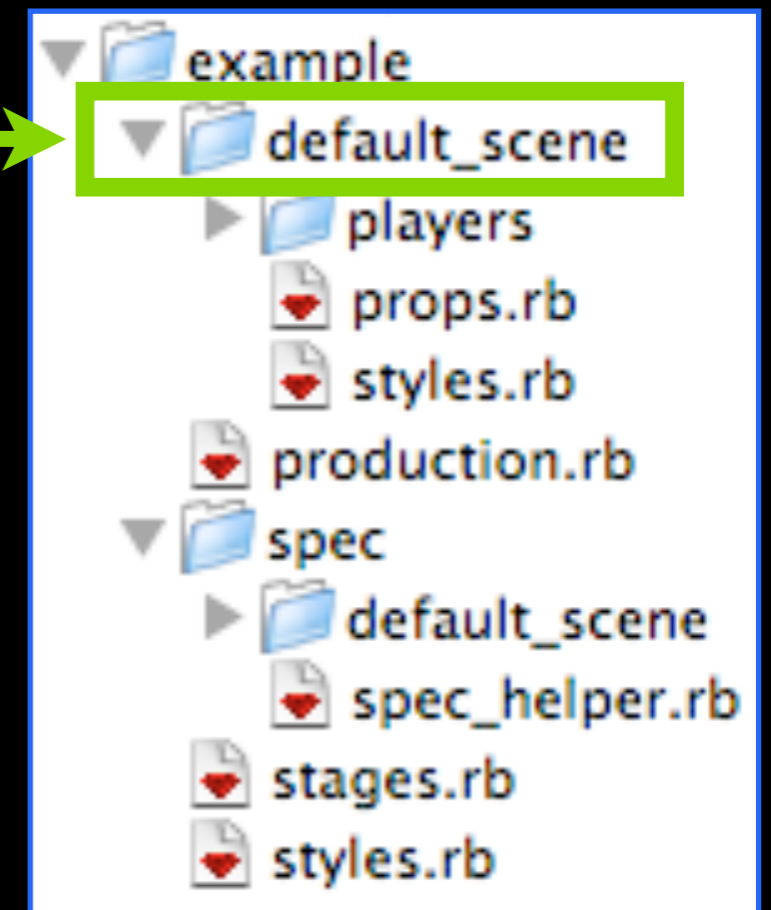
Naming conventions

styles.rb

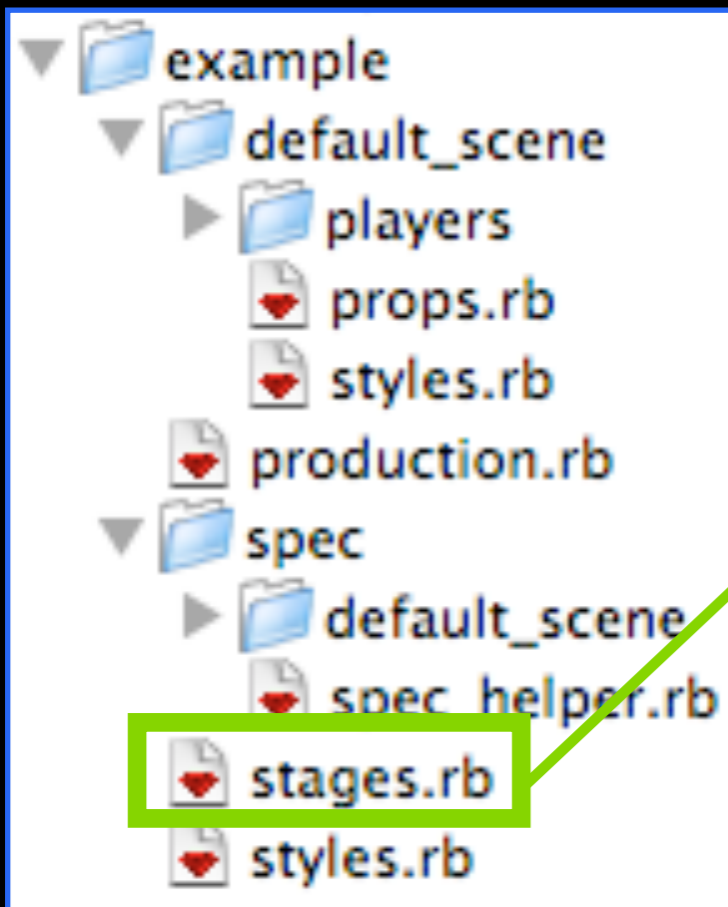
```
1 default_scene {  
2   background_color :black  
3   horizontal_alignment :center  
4   vertical_alignment :center  
5   width "100%"  
6   height "100%"  
7 }  
8  
9 root {  
10  text_color :white  
11  font_size 18  
12 }
```

props.rb

```
1 root :text => "This is the Default Scene scene."
```



stages.rb / Stage DSL



```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```

- Define and configure stages (windows)
- All Ruby Code
- key:value based DSL

Stage Name

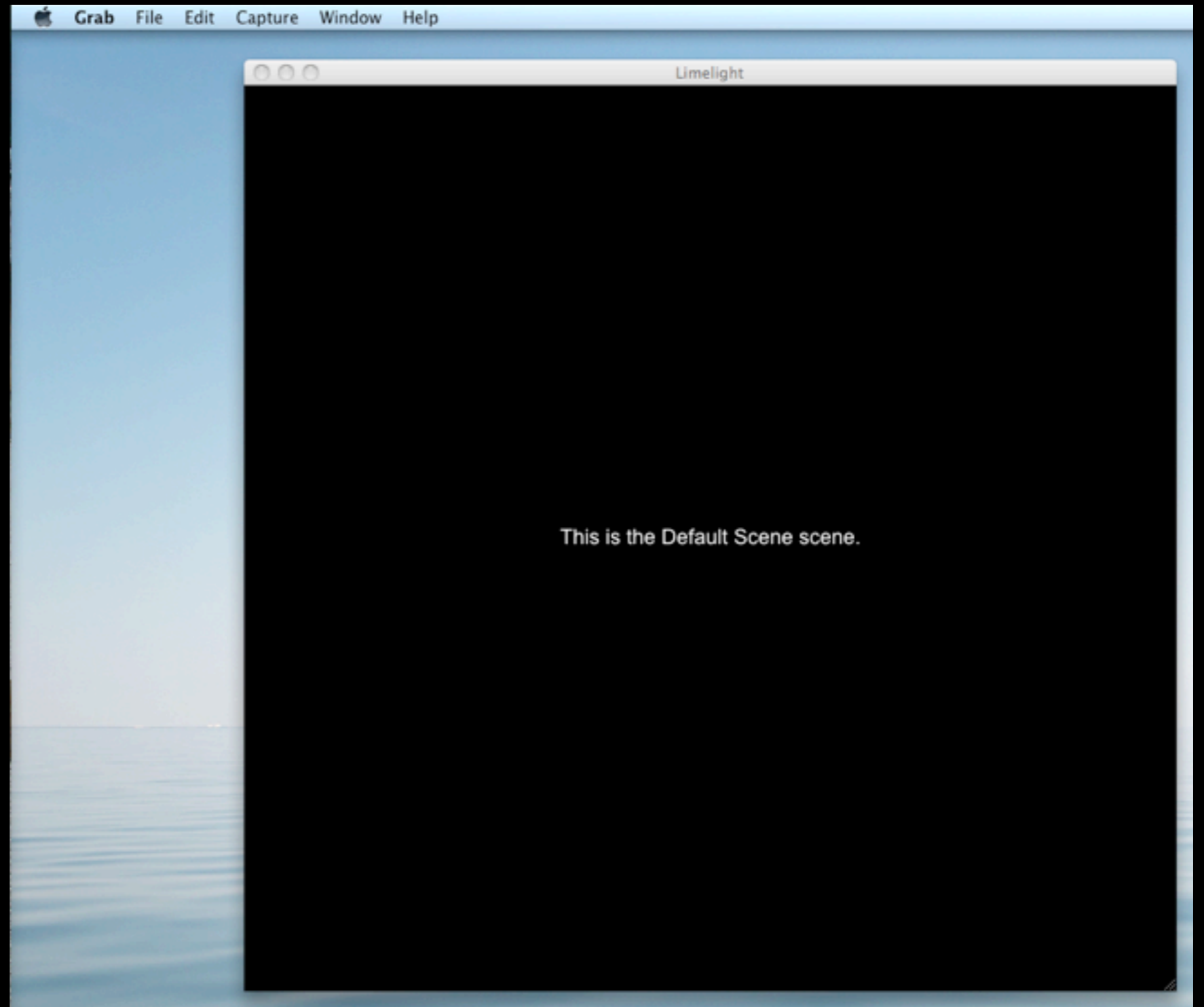
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

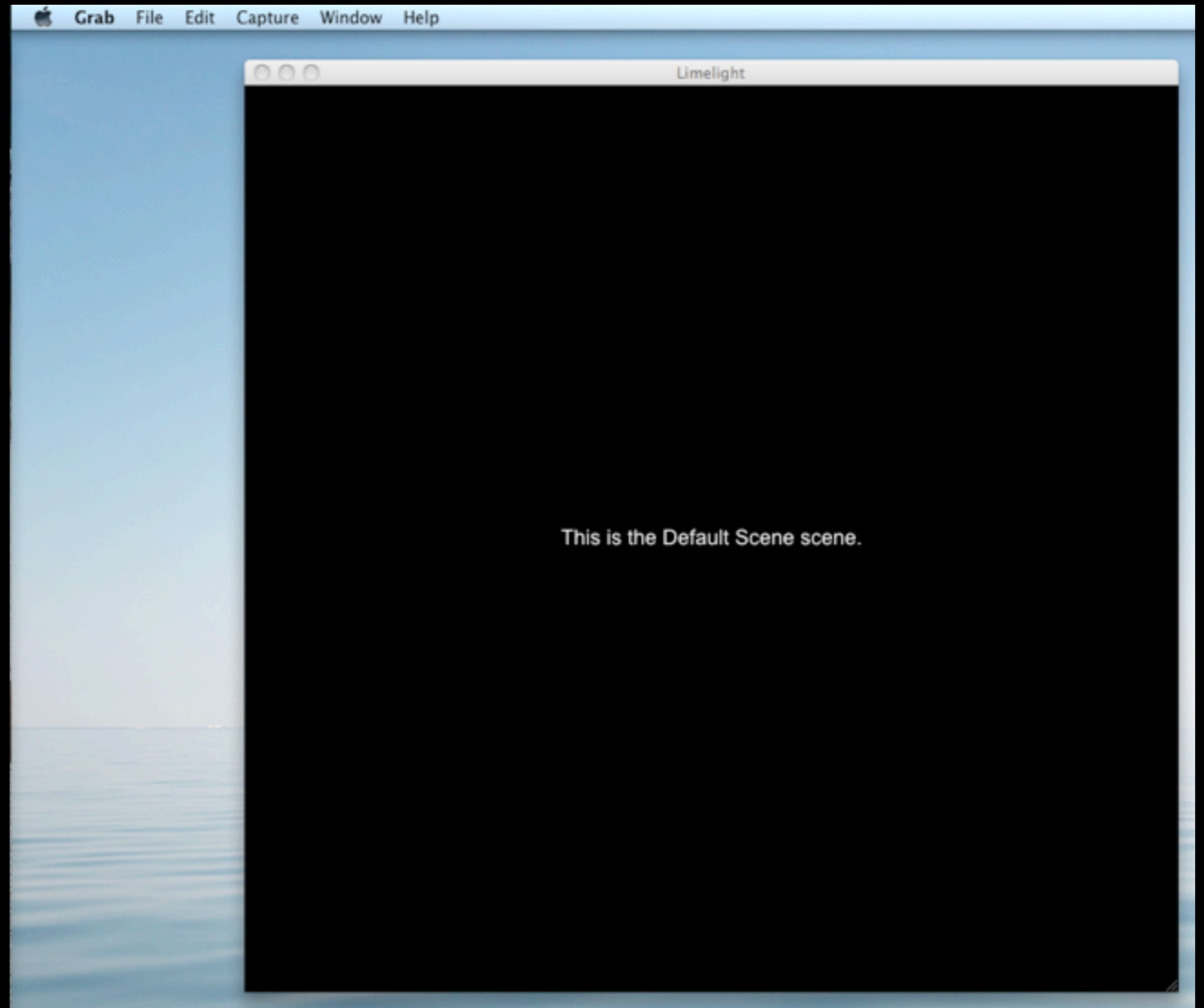
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

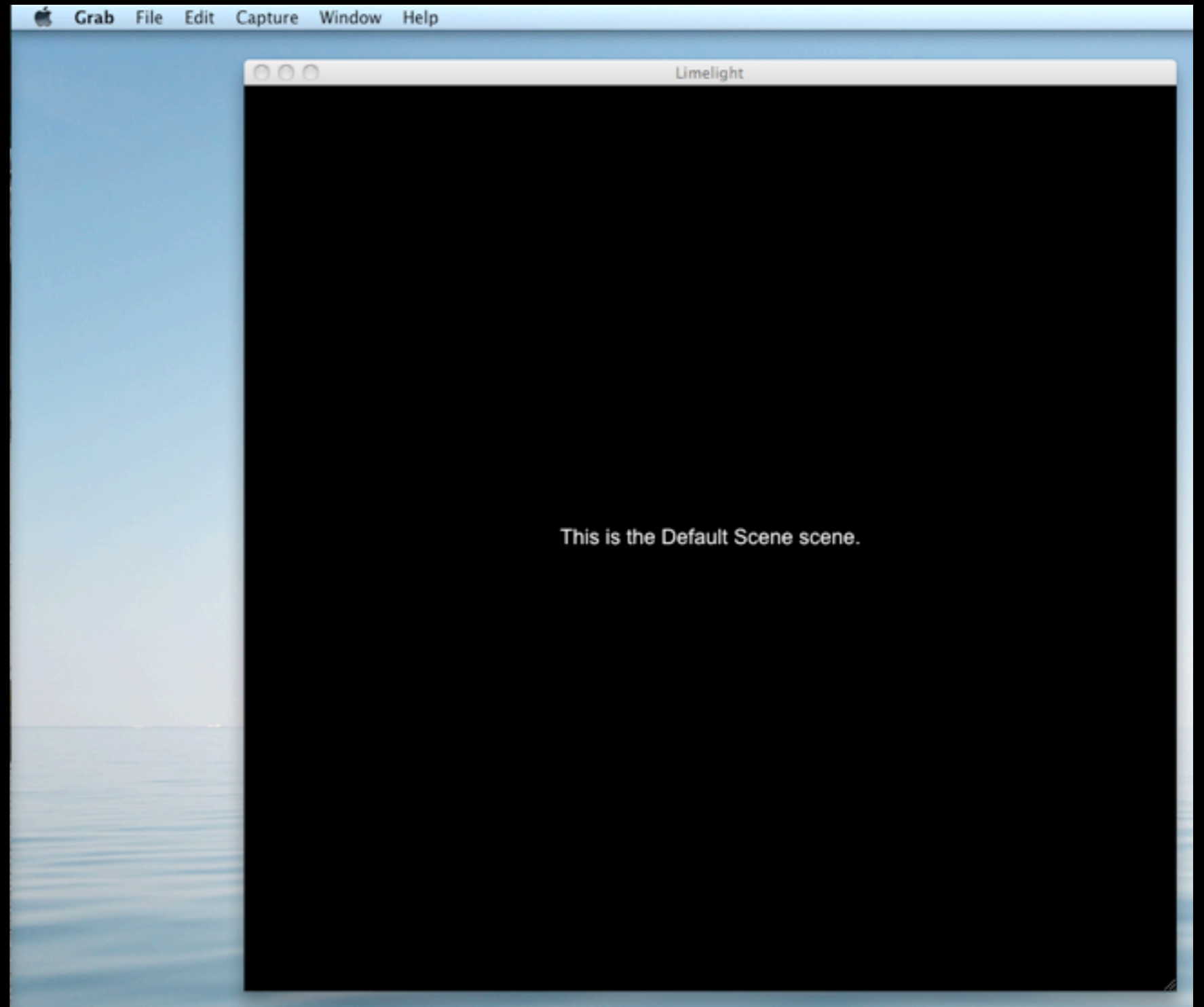
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

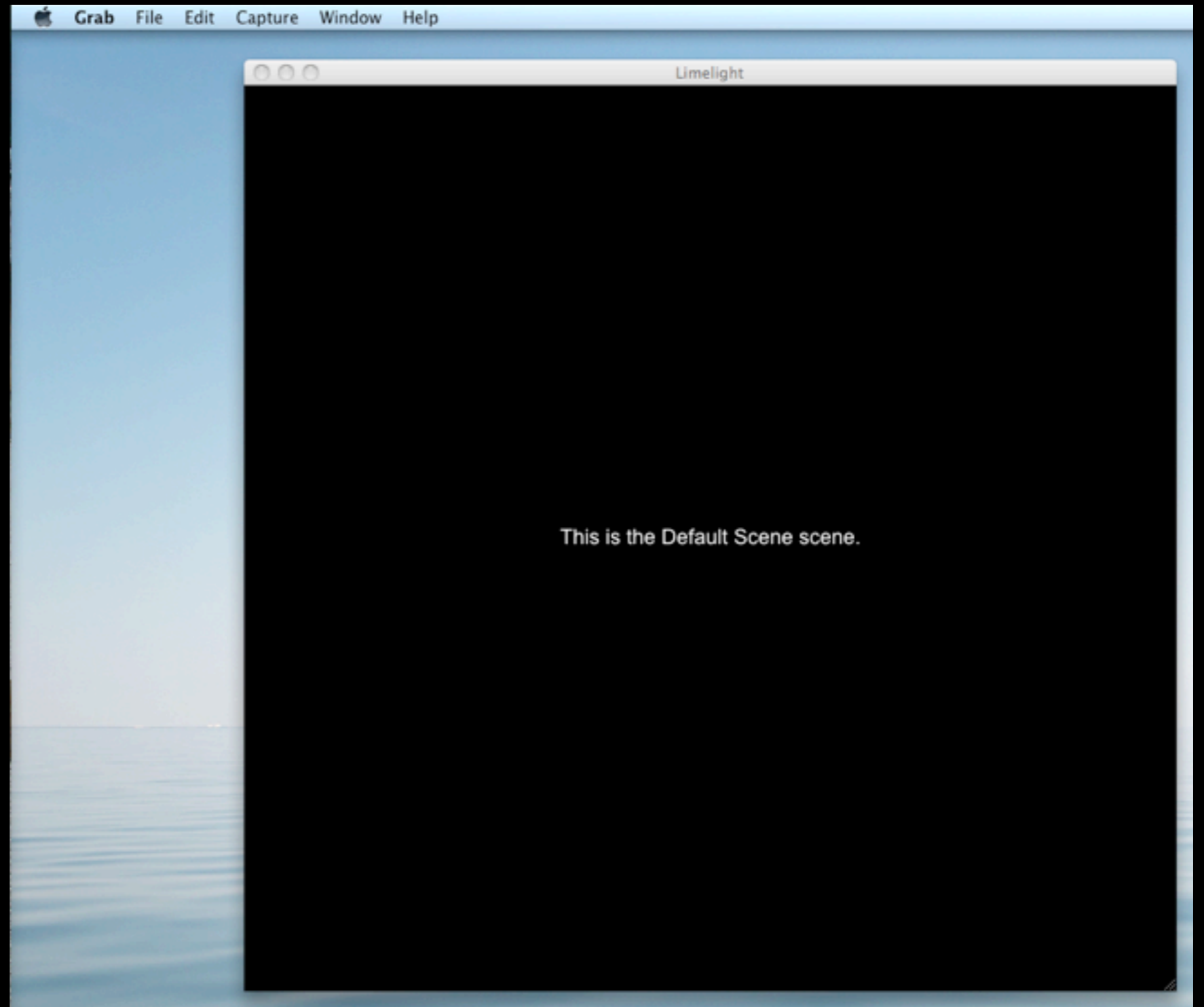
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

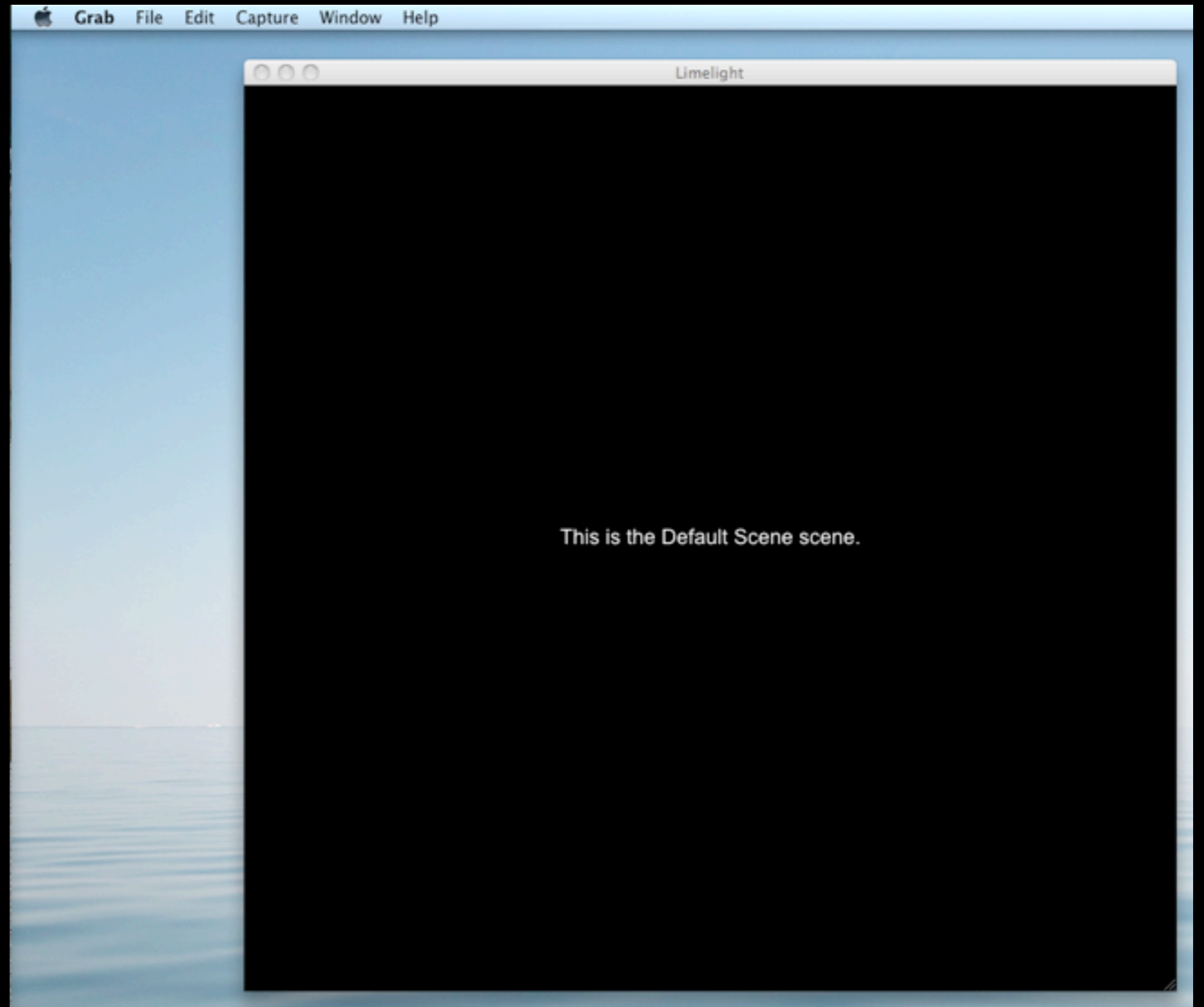
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

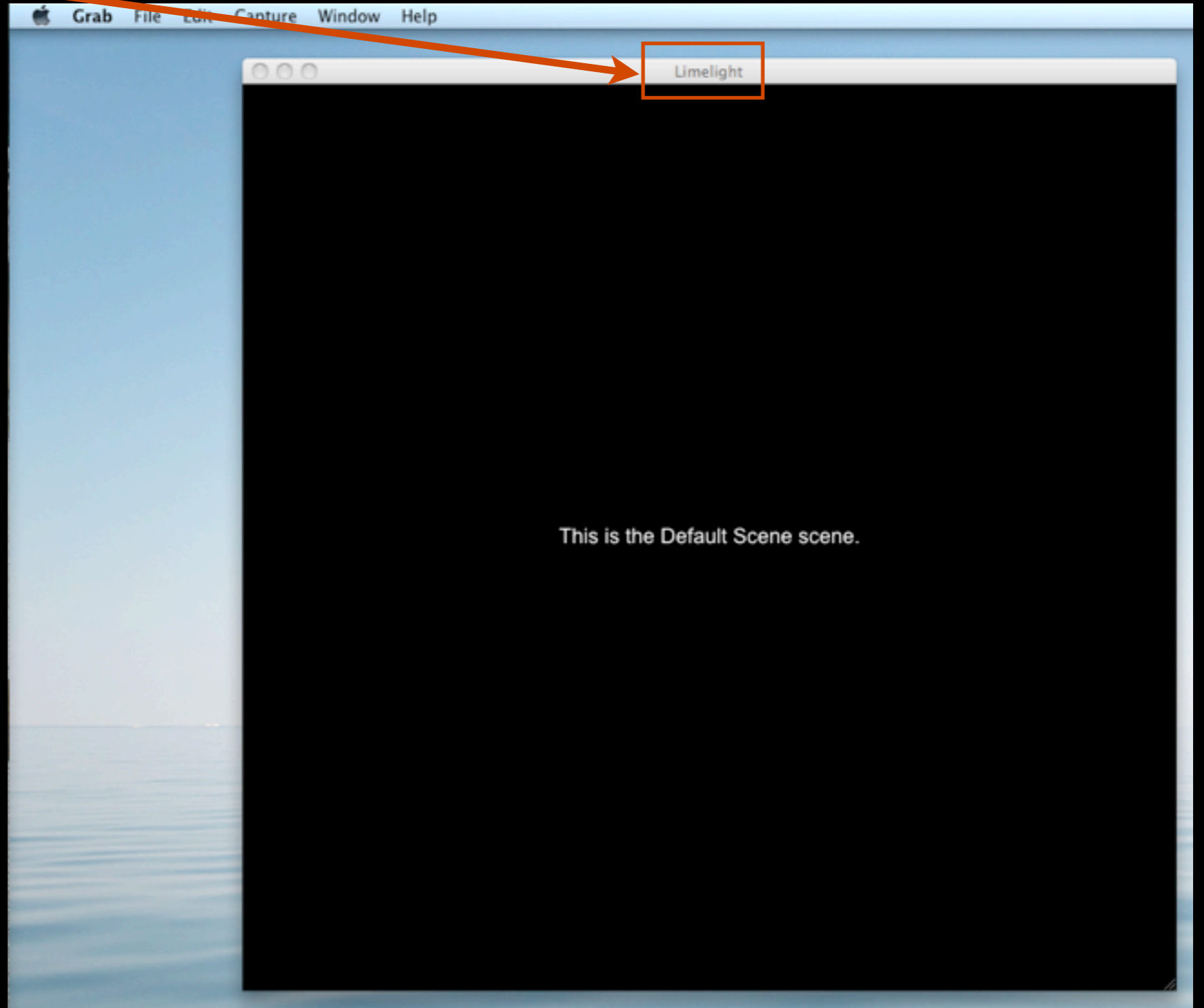
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

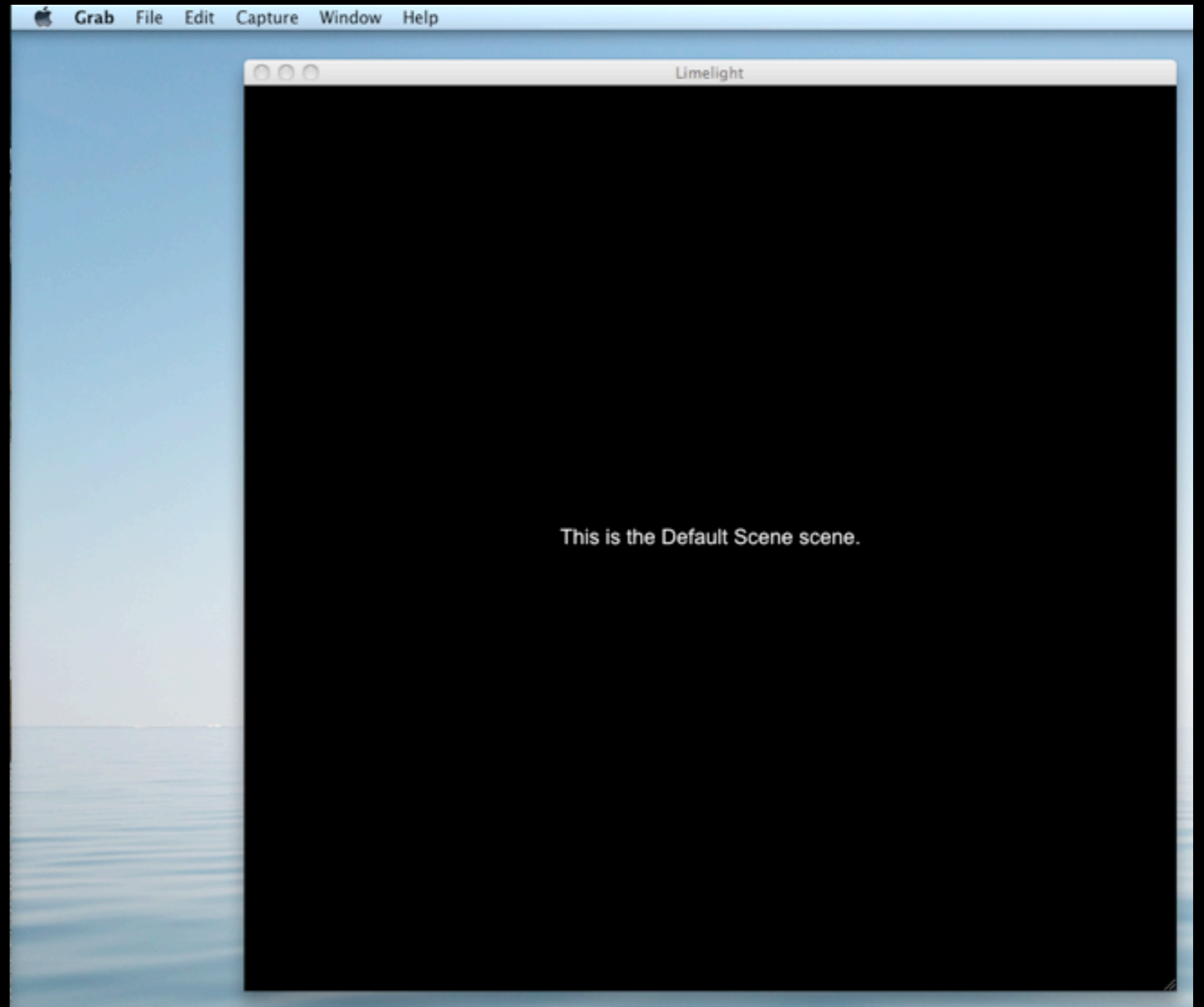
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

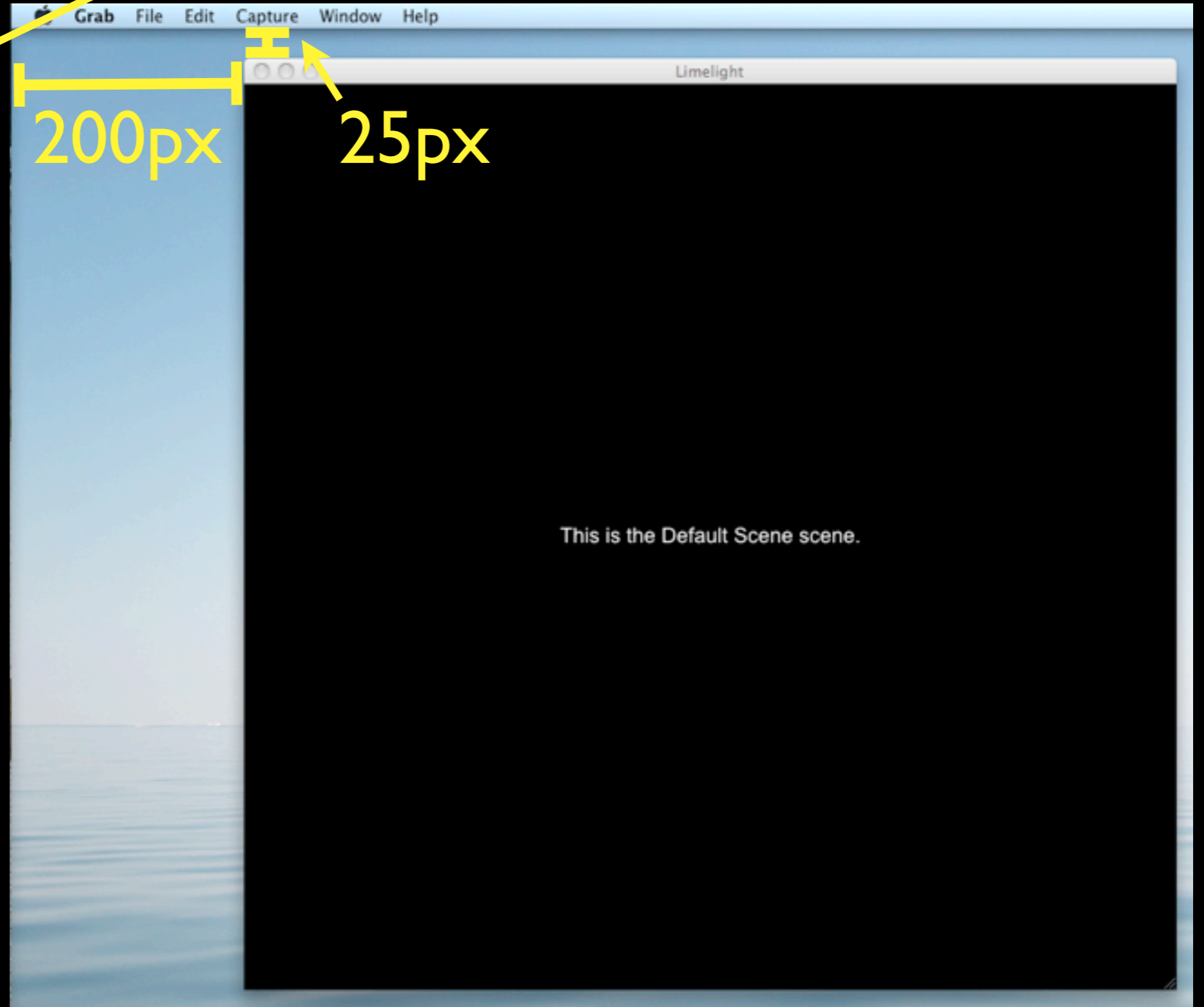
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

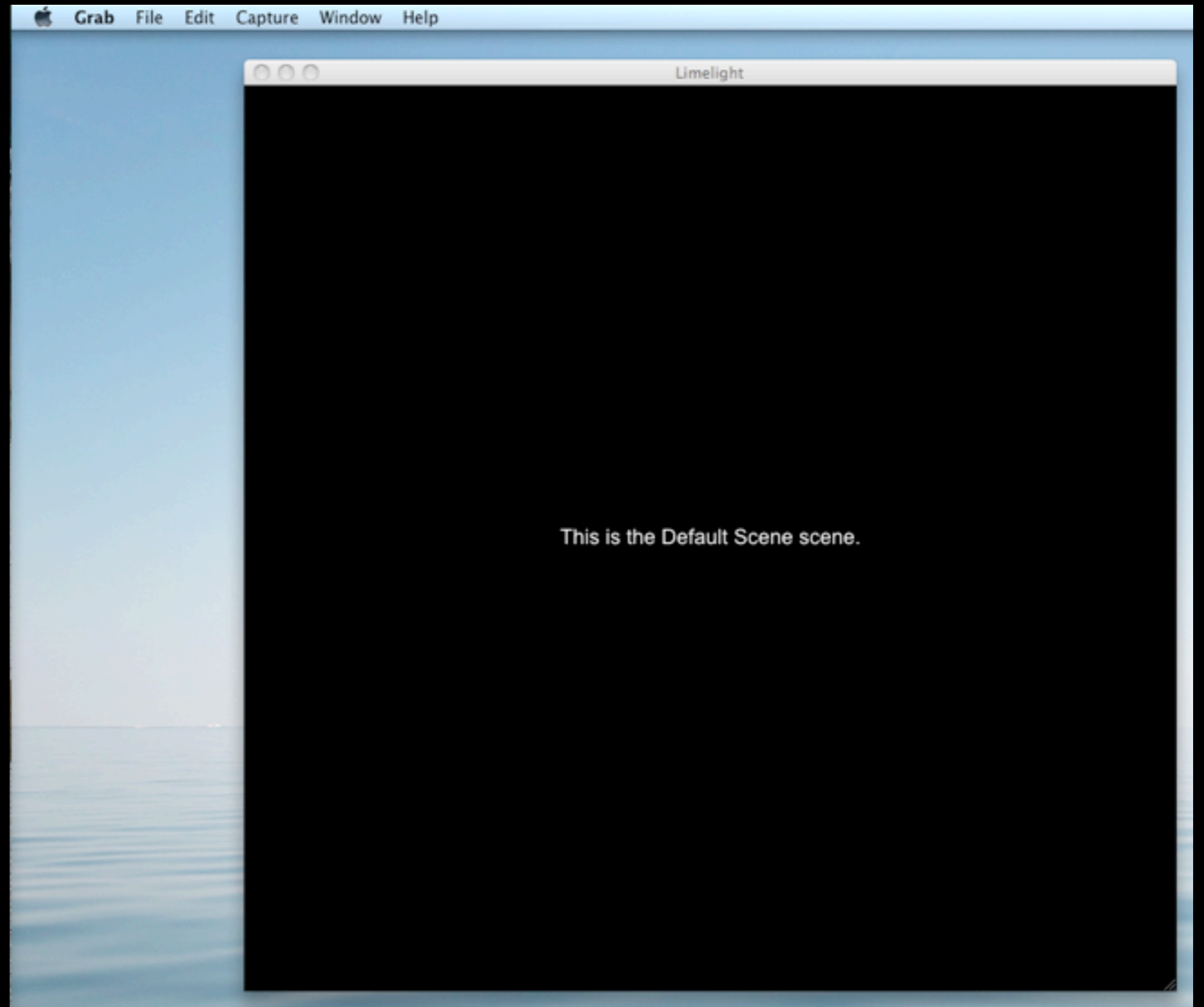
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

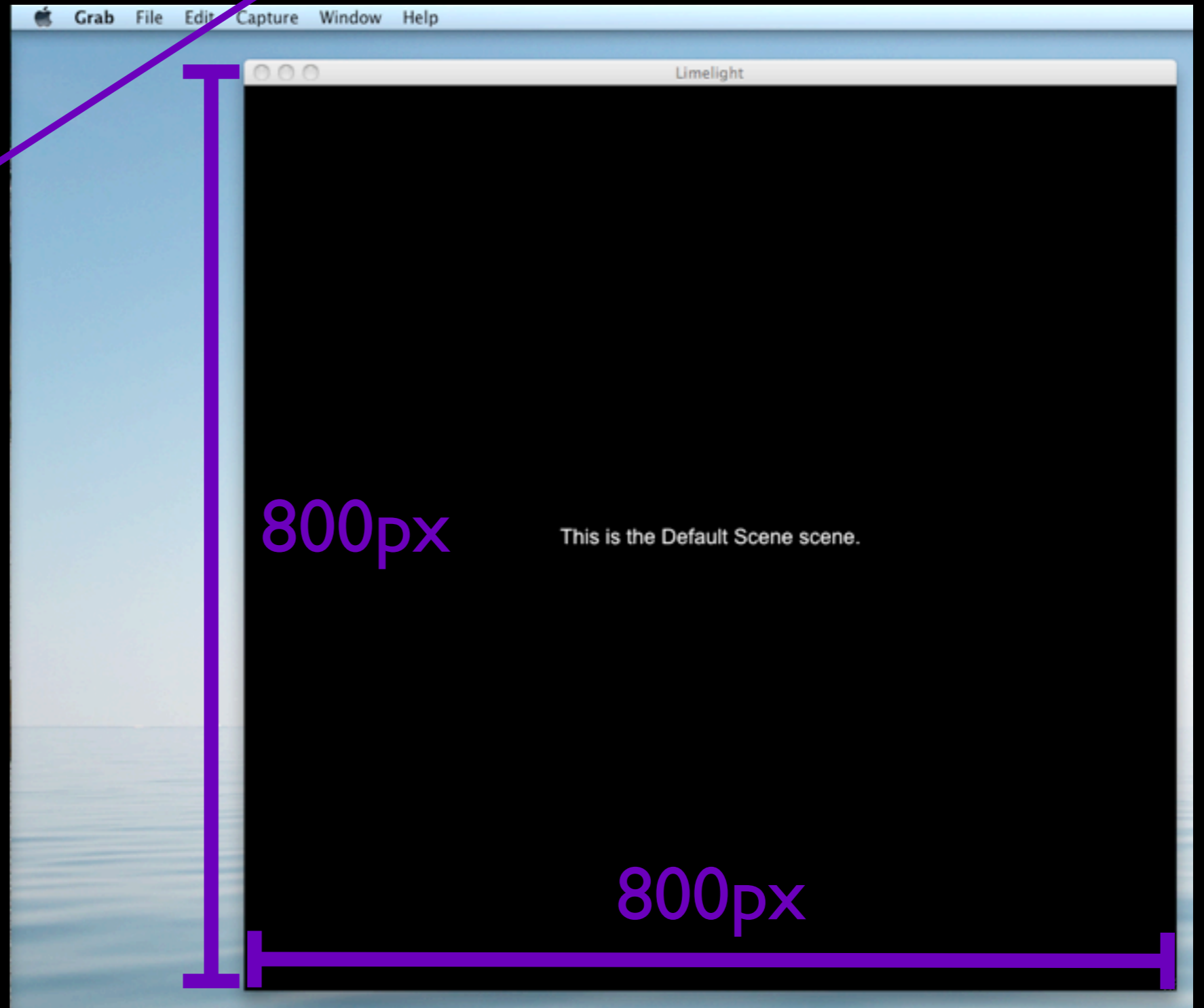
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```



Stage Name

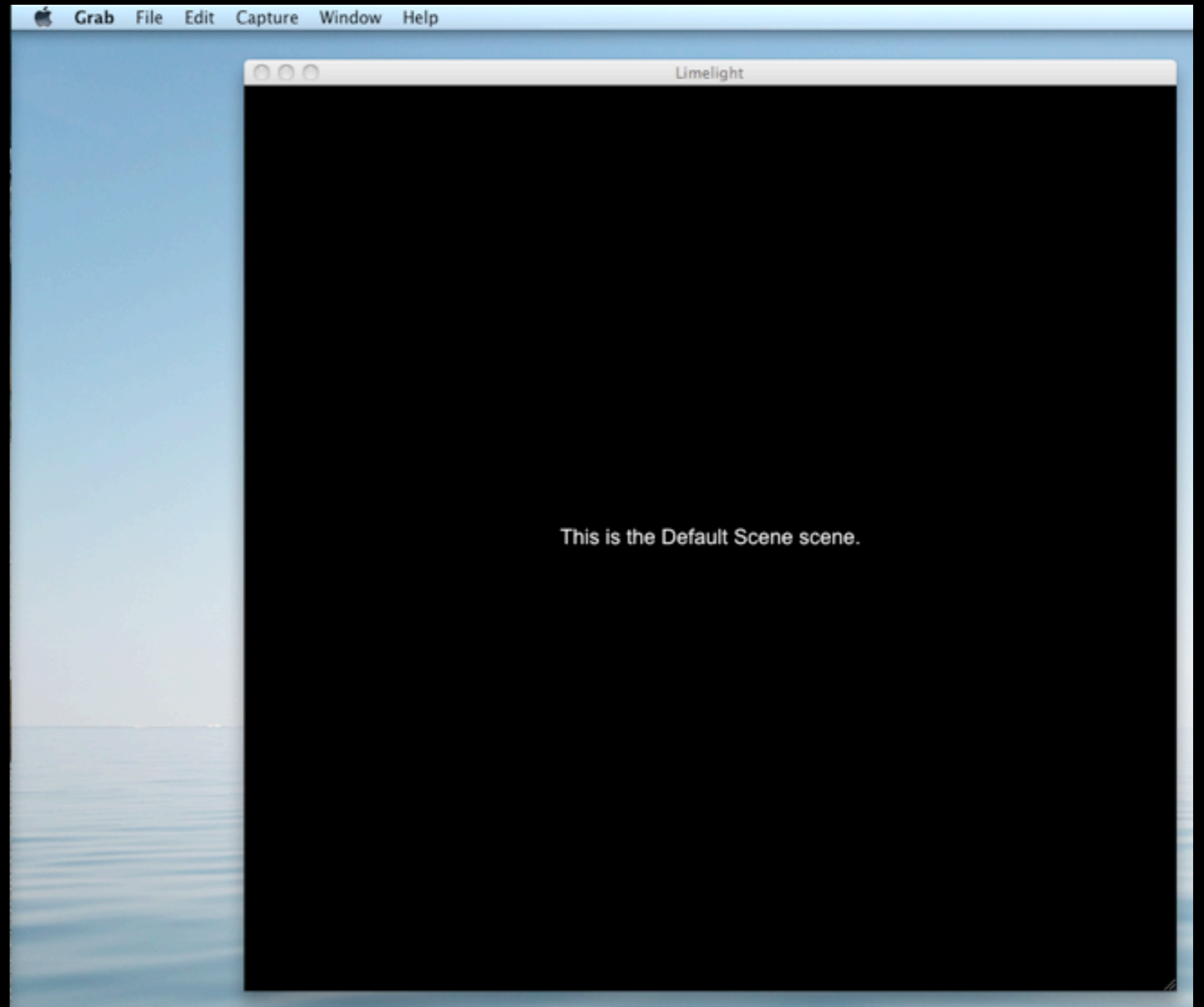
Default Scene

Title

Location

Size

```
1 stage "default" do
2   default_scene "default_scene"
3   title "Limelight"
4   location [200, 25]
5   size [800, 800]
6 end
```

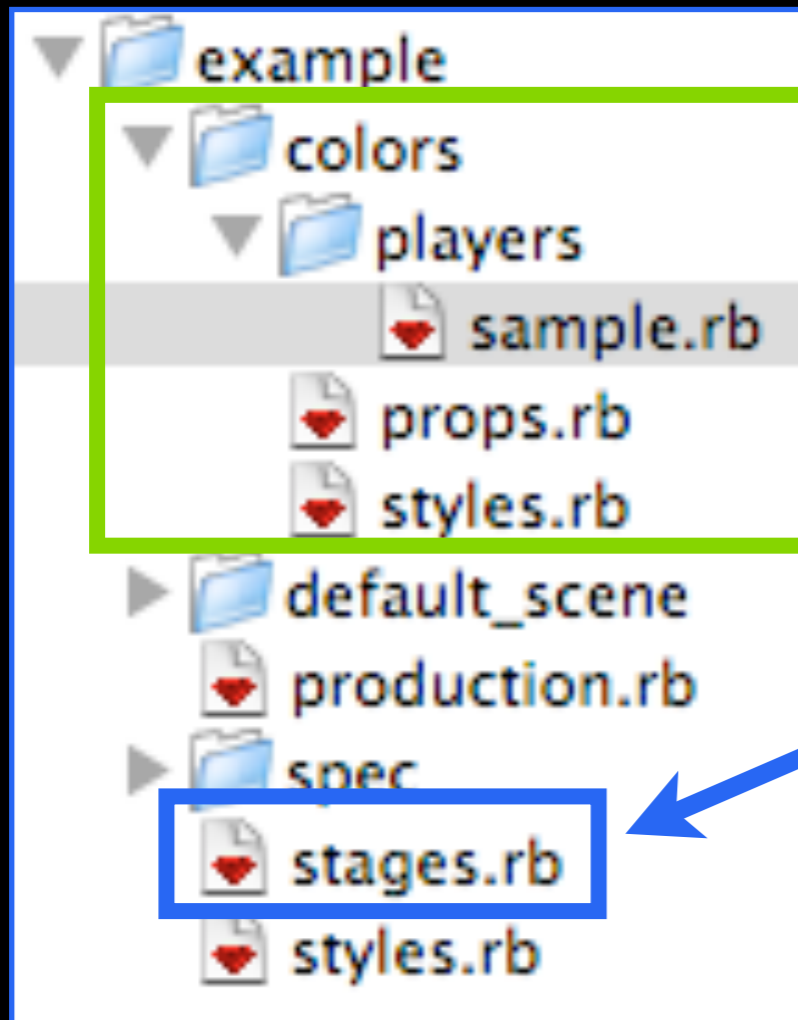




All the World's a
Stage and We are
Merely **Players**

Copyright 2009 Micah Martin

Colors Scene



New Scene
named "colors"

Changes to stages.rb

```
9 stage "colors" do
10   default_scene "colors"
11   title "Colors"
12   location :center, :center
13   size [700, 400]
14 end
```

Colors Code

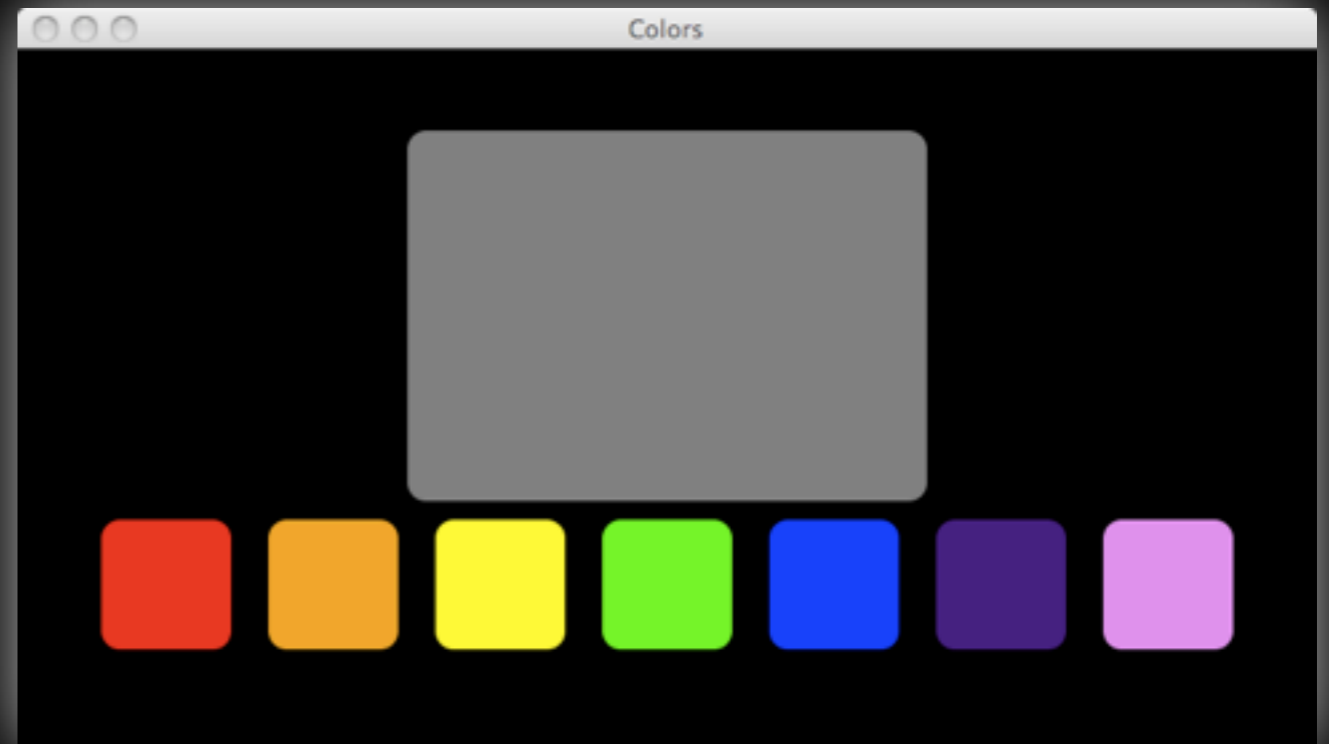
props.rb

```
1 selection :id => "selection"
2 %w{ red orange yellow green blue indigo violet }.each do |color|
3   sample :background_color => color
4 end
```

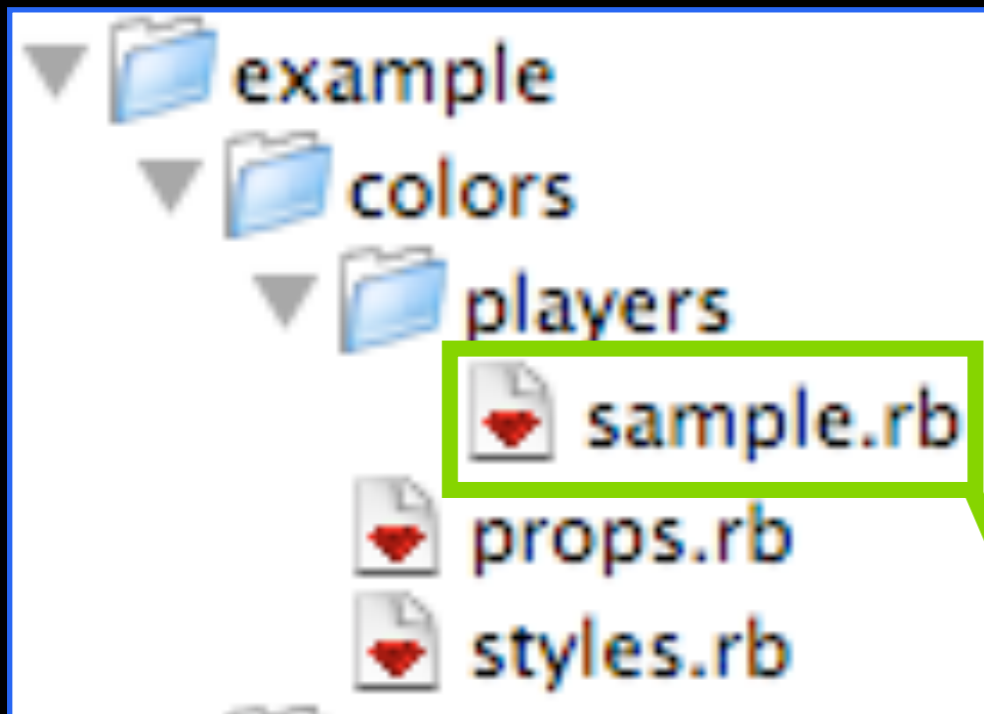
styles.rb

```
1 selection {
2   width "100%"
3   left_margin "30%"
4   right_margin "30%"
5   height 200
6   background_color :gray
7   rounded_corner_radius 10
8 }
9
10 sample {
11   width 90
12   height 90
13   rounded_corner_radius 10
14   margin 10
15 }
```

on screen appearance



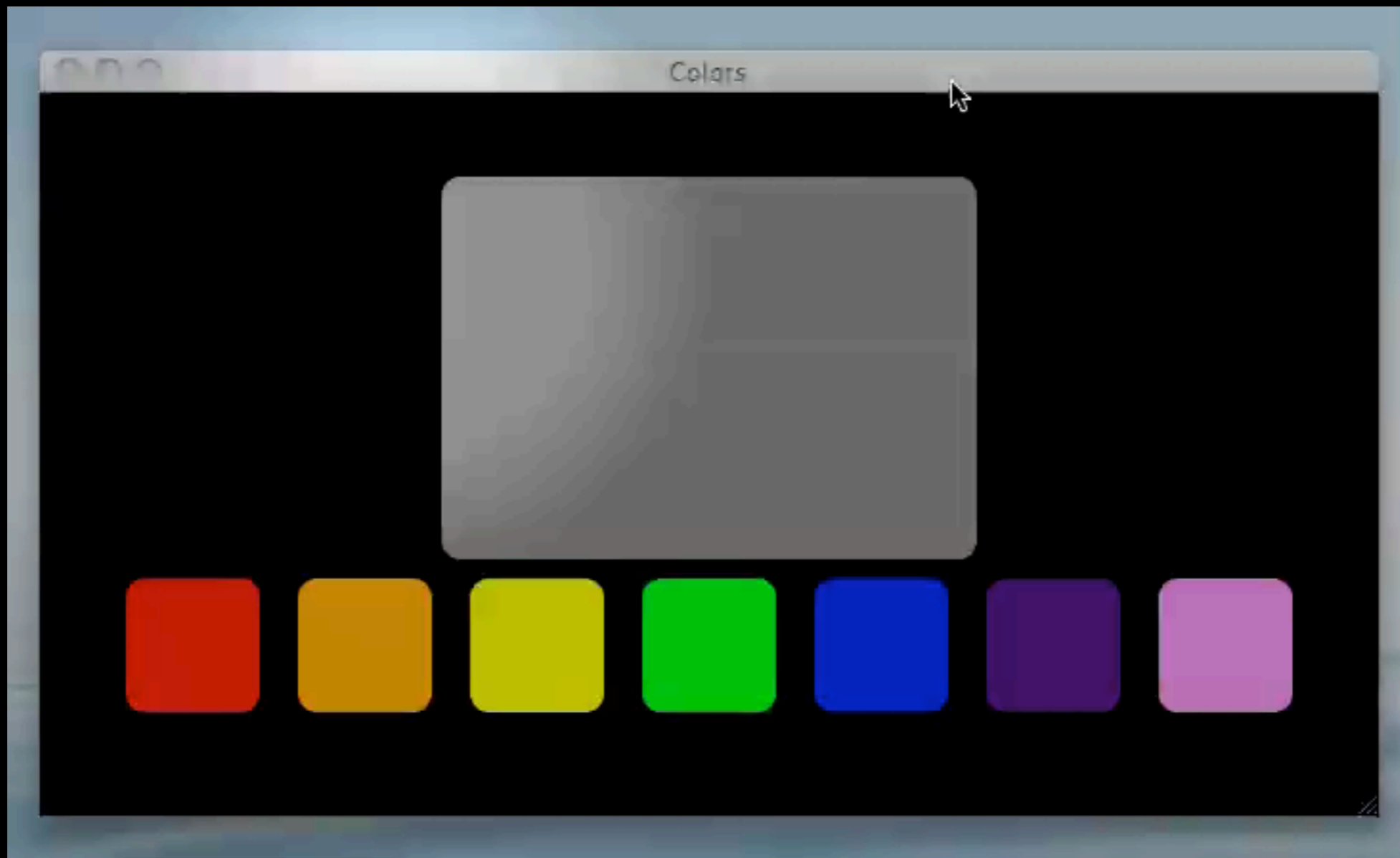
Adding Player



remember.. the colored props are named "sample"

```
1 module Sample
2   def mouse_clicked(e)
3     selection = scene.find("selection")
4     color = self.style.background_color
5     selection.style.background_color = color
6   end
7 end
```

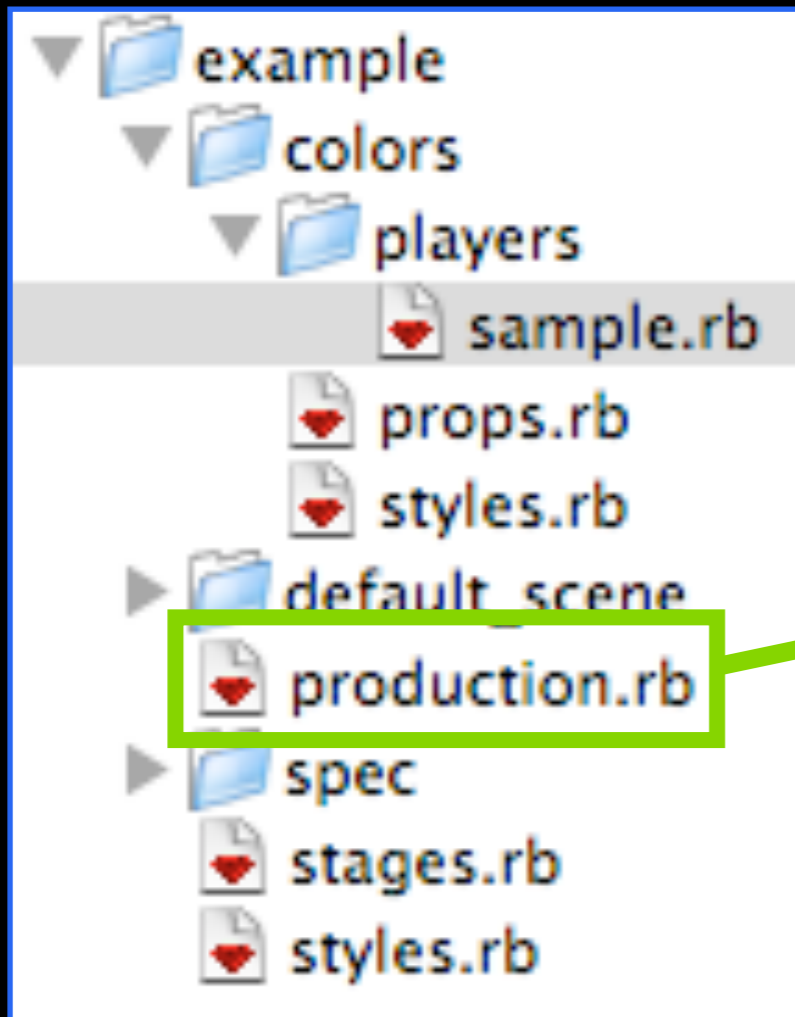

Colors in Action



Prop Hooks

- mouse_clicked
- mouse_entered
- mouse_exited
- mouse_pressed
- mouse_released
- mouse_dragged
- mouse_moved
- key_typed
- key_pressed
- key_released

Production Hooks



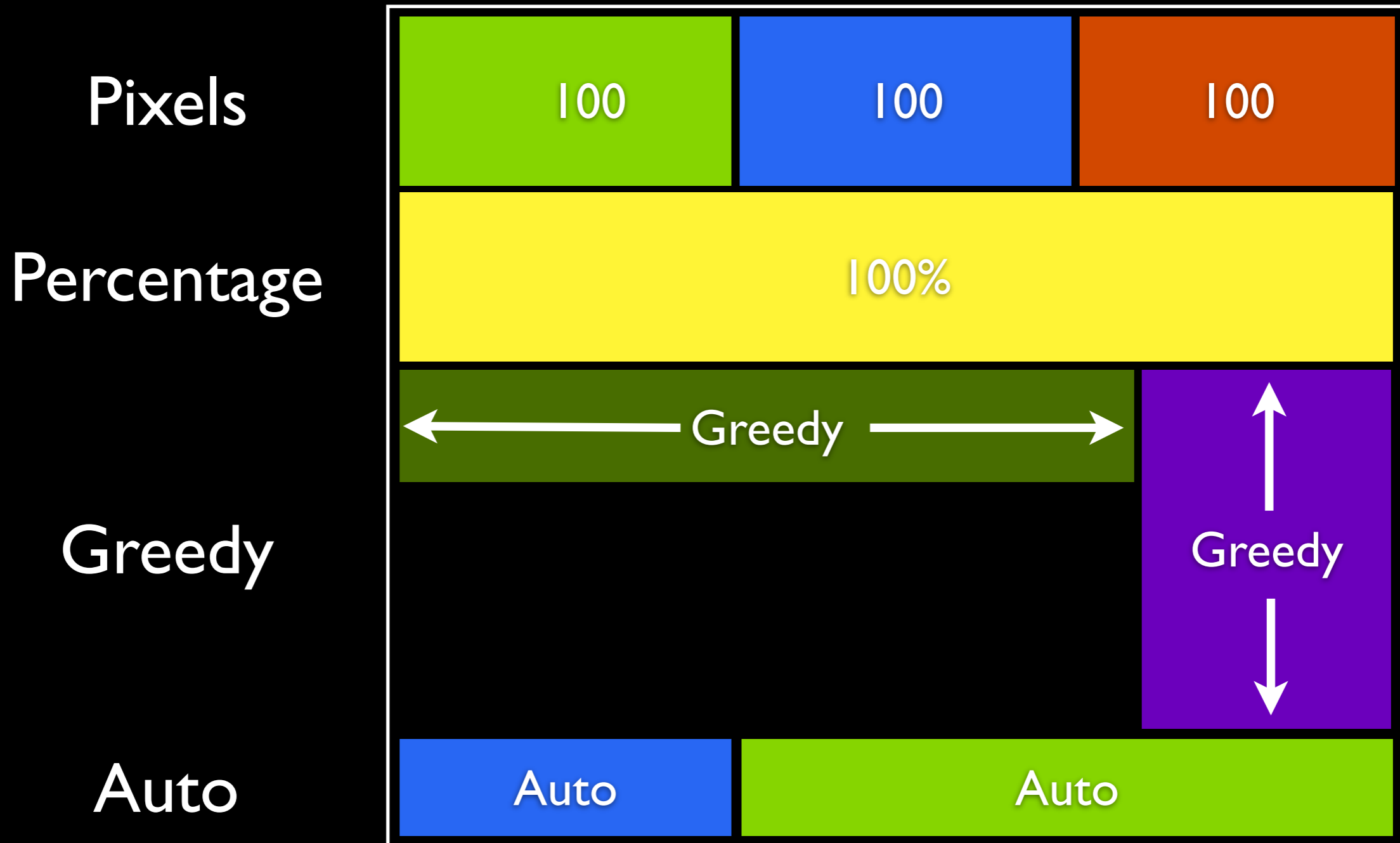
```
1 module Production
2 #   def production_opening
3 #   end
4 #
5 #   def production_loaded
6 #   end
7 #
8 #   def production_opened
9 #   end
10 #
11 #   def allow_close?
12 #     return true
13 #   end
14 #
15 #   def production_closing
16 #   end
17 #
18 #   def production_closed
19 #   end
20 end
```



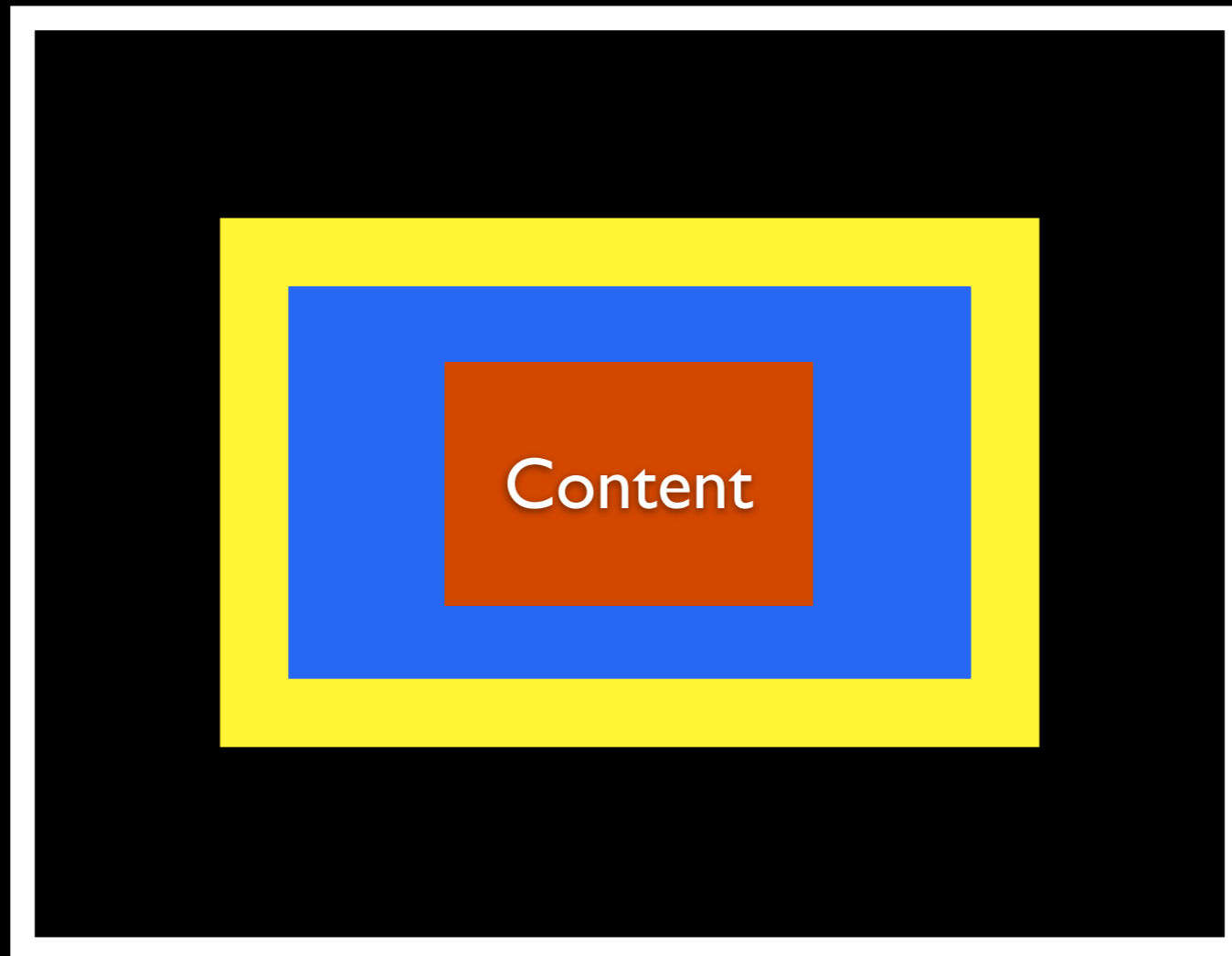
Limelight Styling

Copyright 2009 Micah Martin

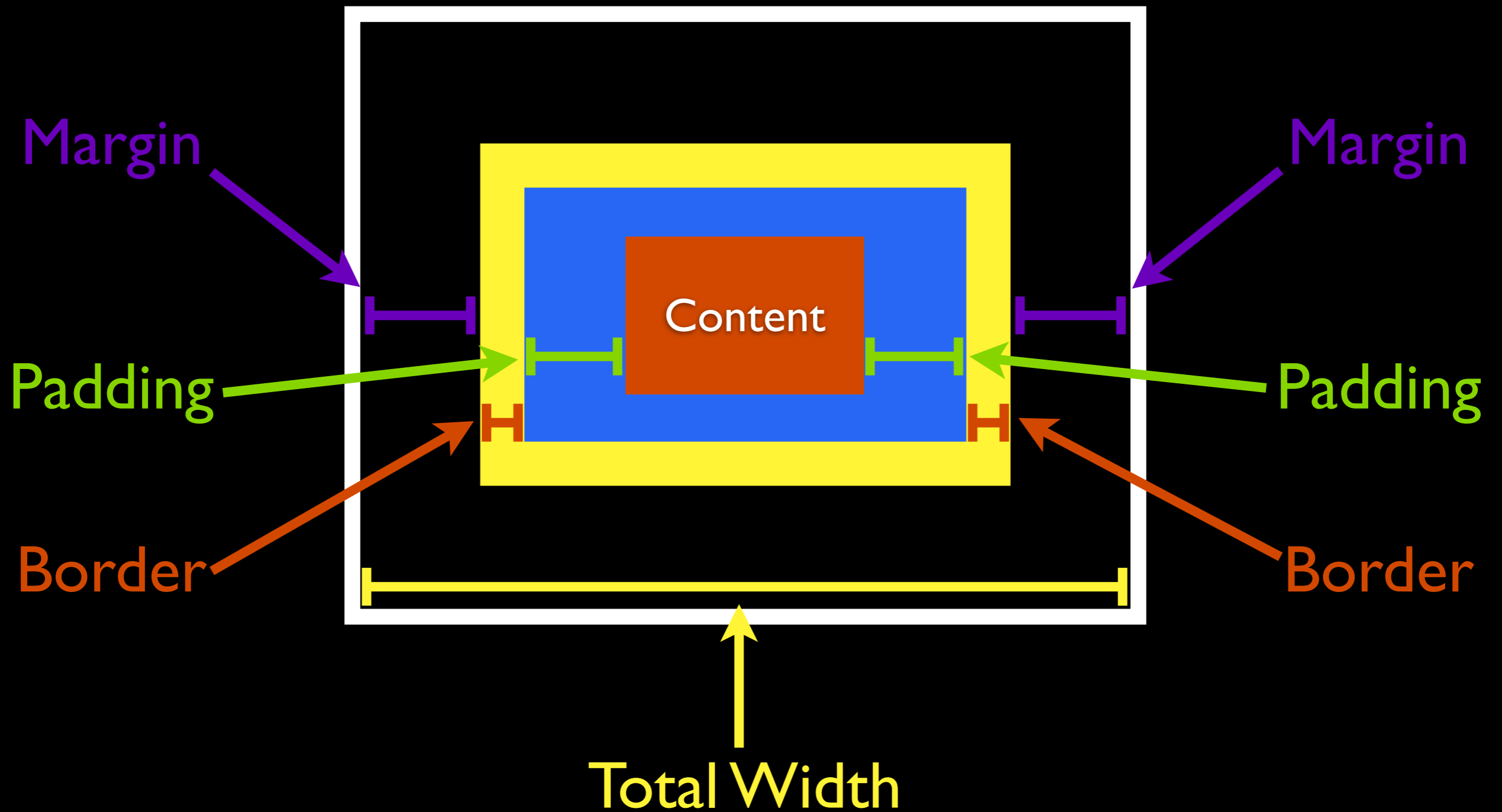
Sizing Props



Insets



Insets



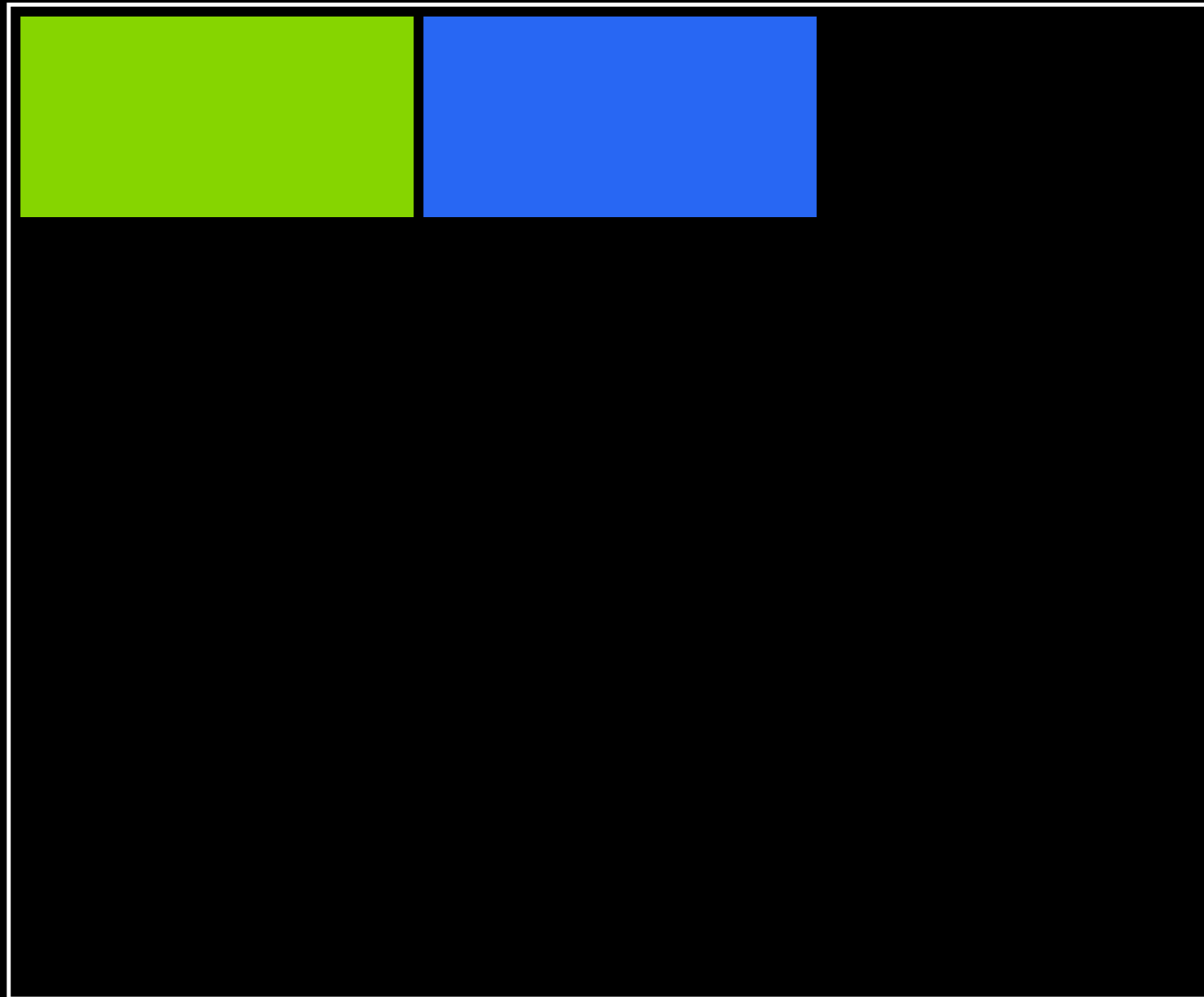
Simple Layout



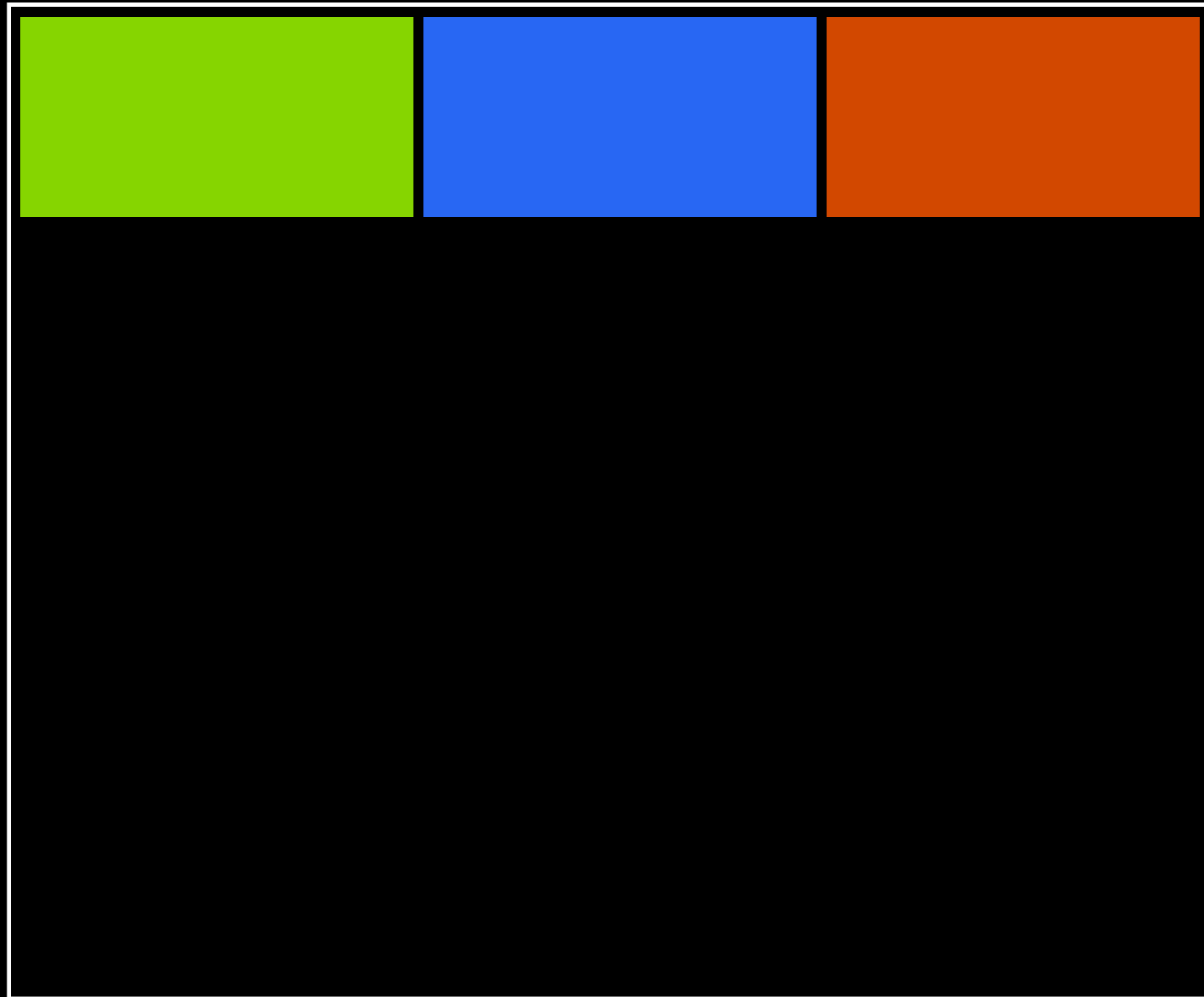
Simple Layout



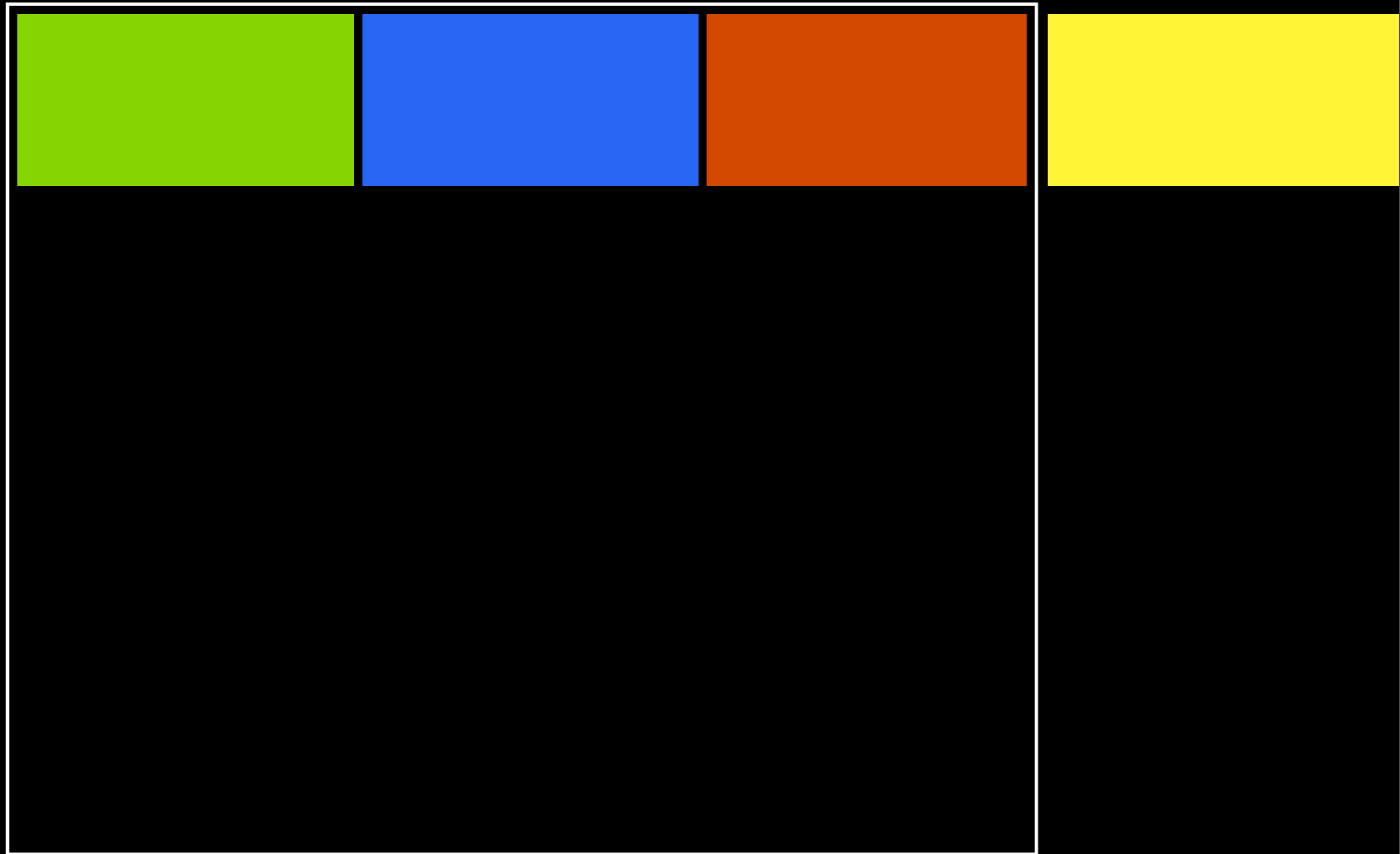
Simple Layout



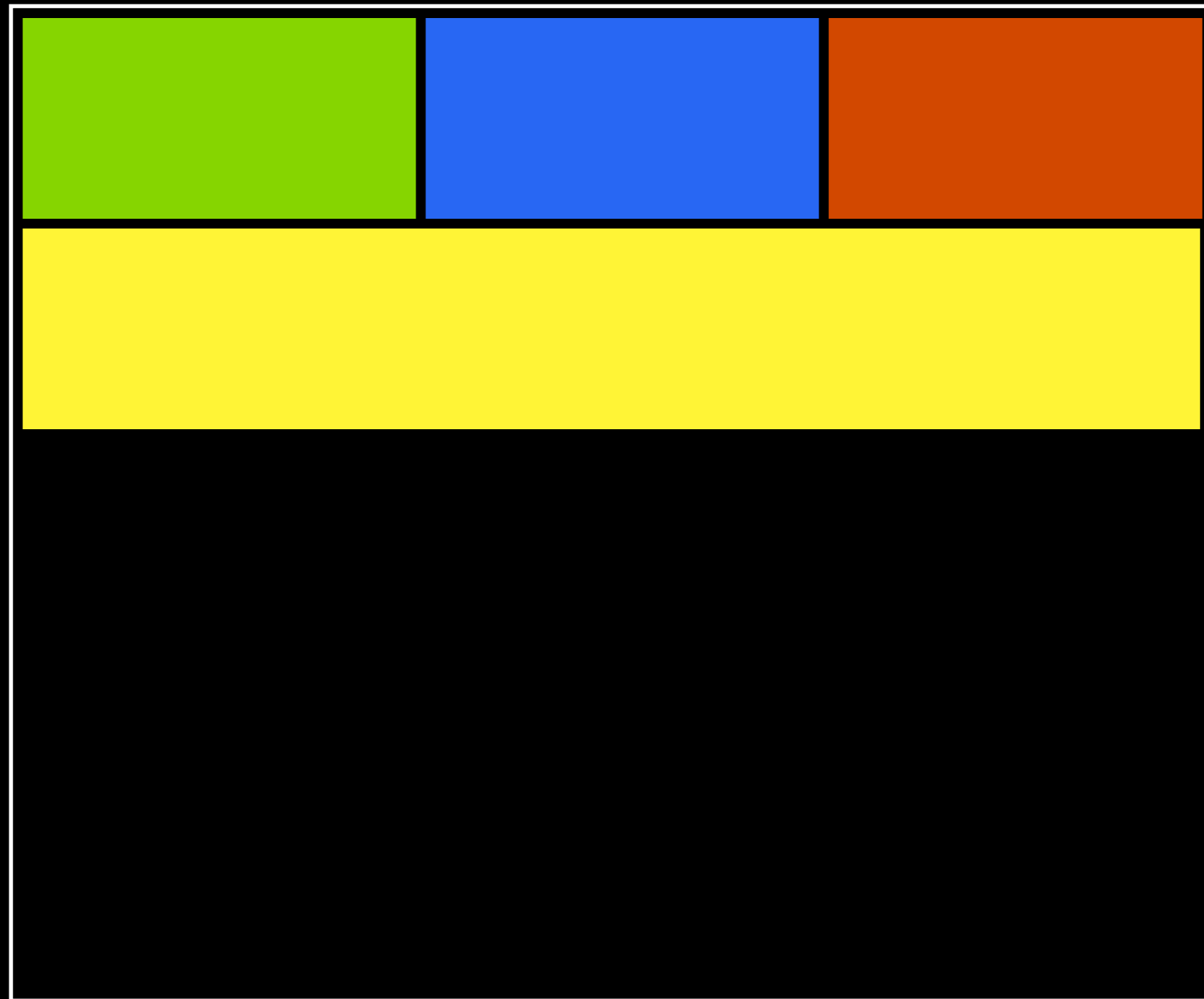
Simple Layout



Simple Layout



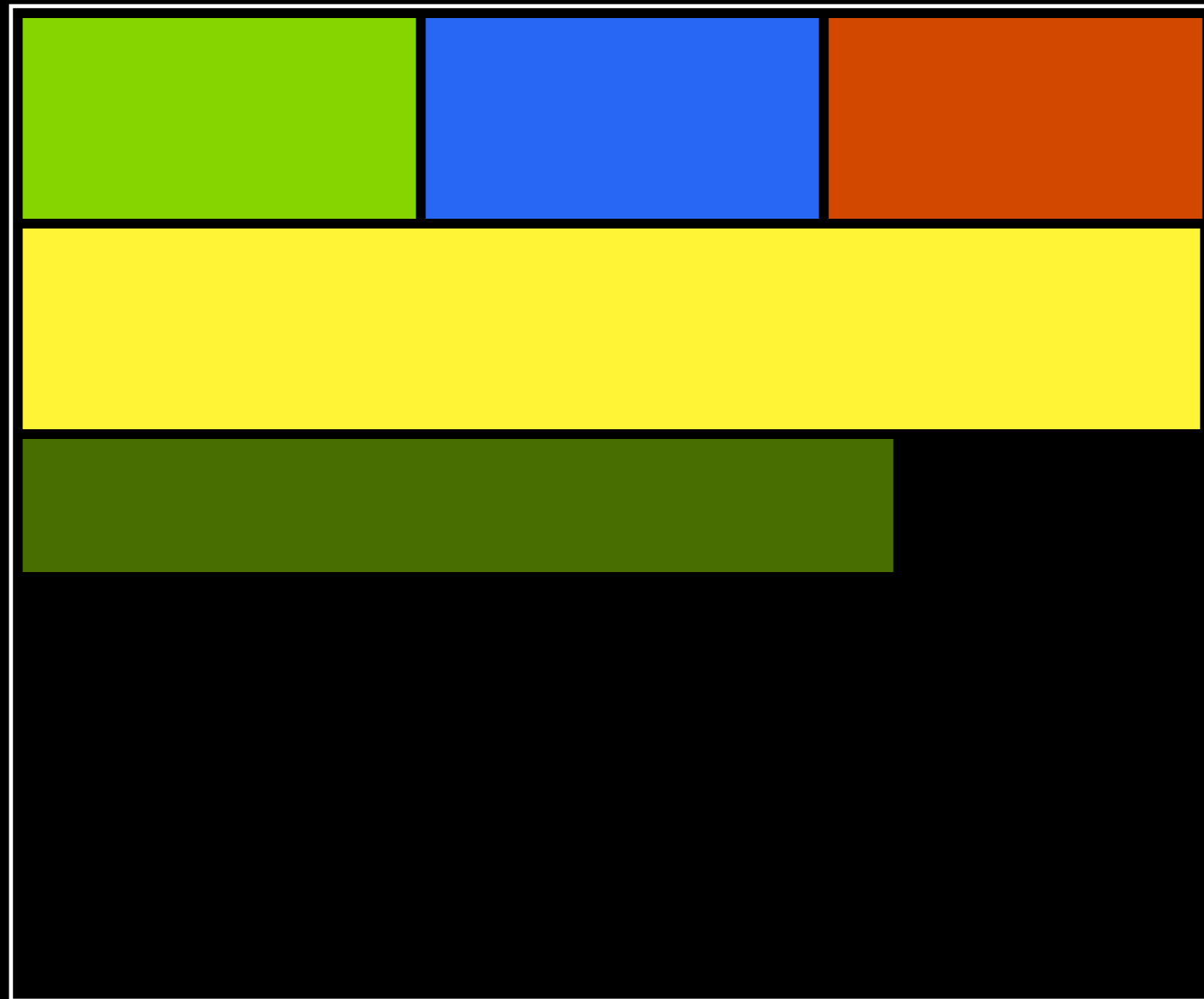
Simple Layout



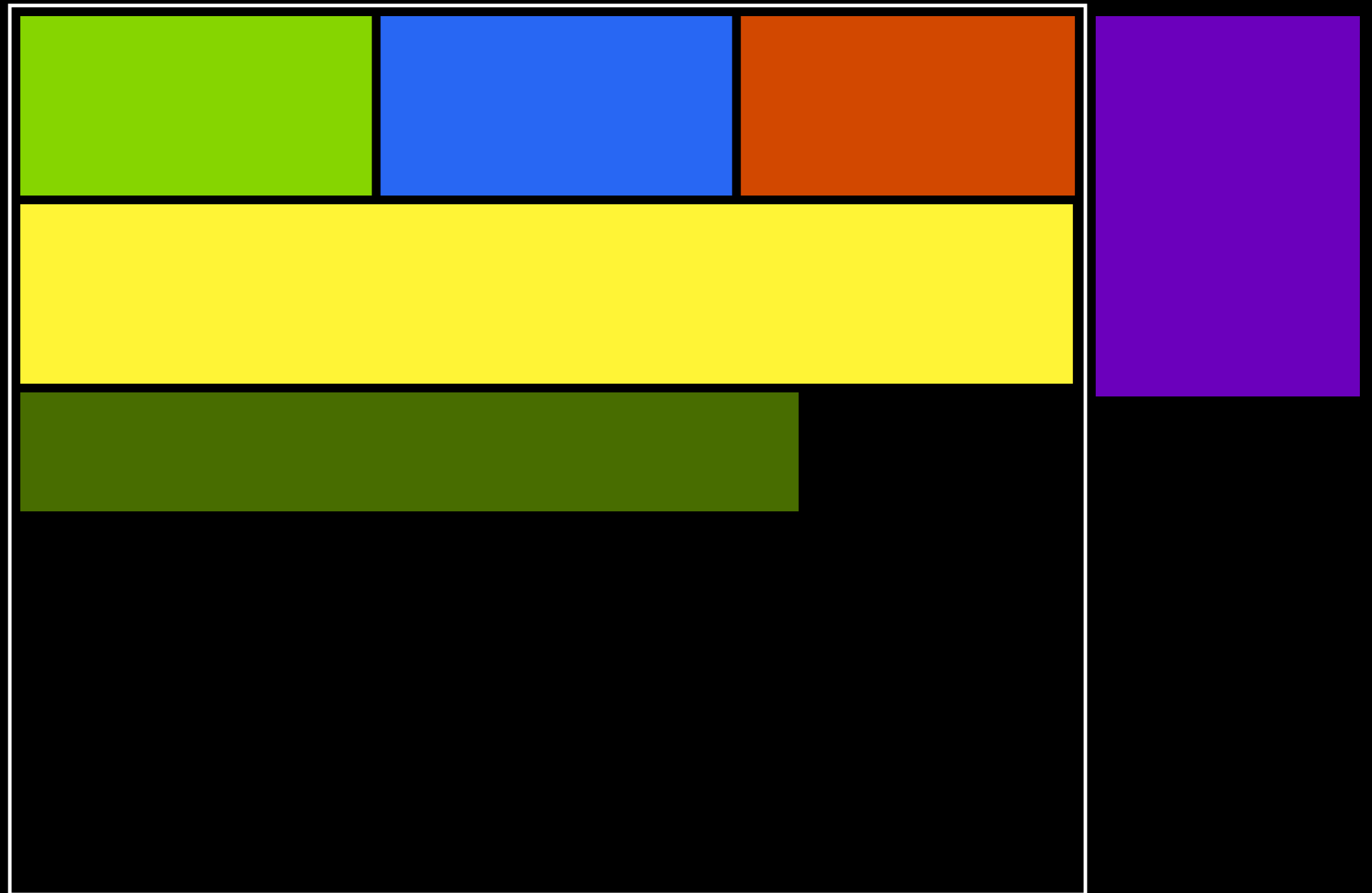
Simple Layout



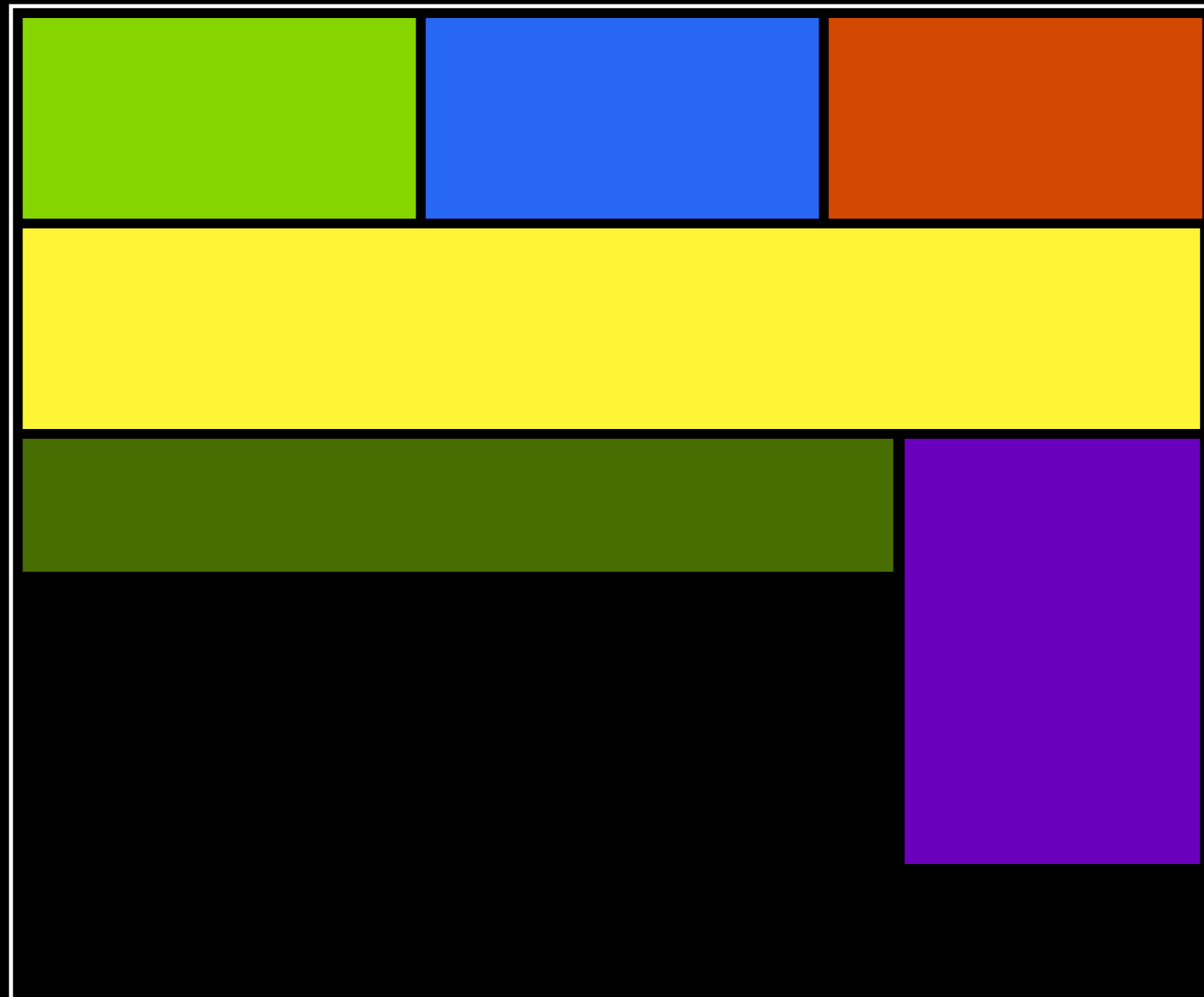
Simple Layout



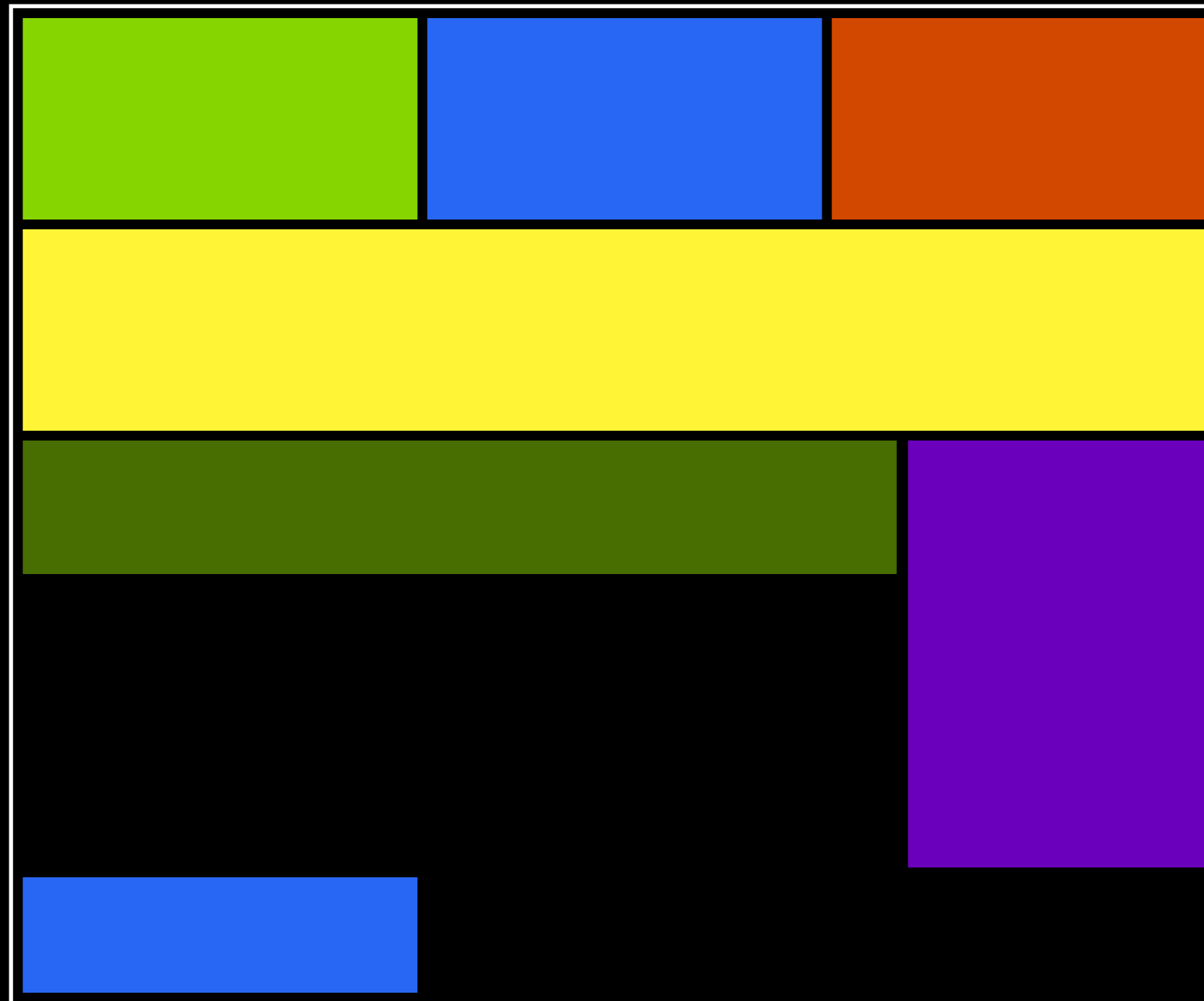
Simple Layout



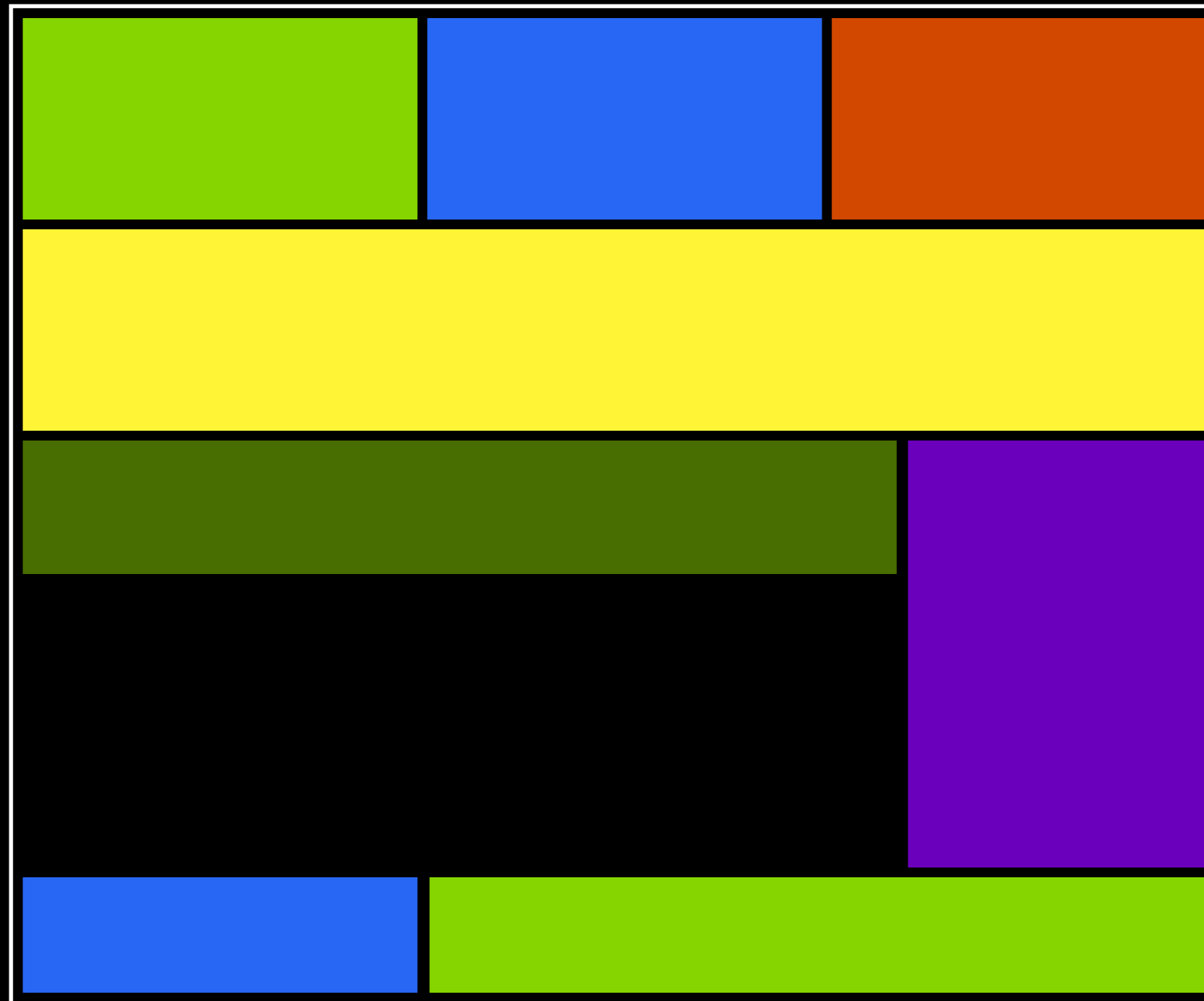
Simple Layout



Simple Layout



Simple Layout



Rounded Corners

radius: 200

radius: 100



radius: 0

radius: 50

Text

Font Face: All Fonts  On Your **SYSTEM** *Are Supported*

Font Size: Make text as **big** (or small) as you want

Font Style: **Bold, Italic, Bold Italic**

Text Color: **Any color of the rainbow, and more**

Backgrounds

Background Image



Solid Color Background

2 Color Gradient Background

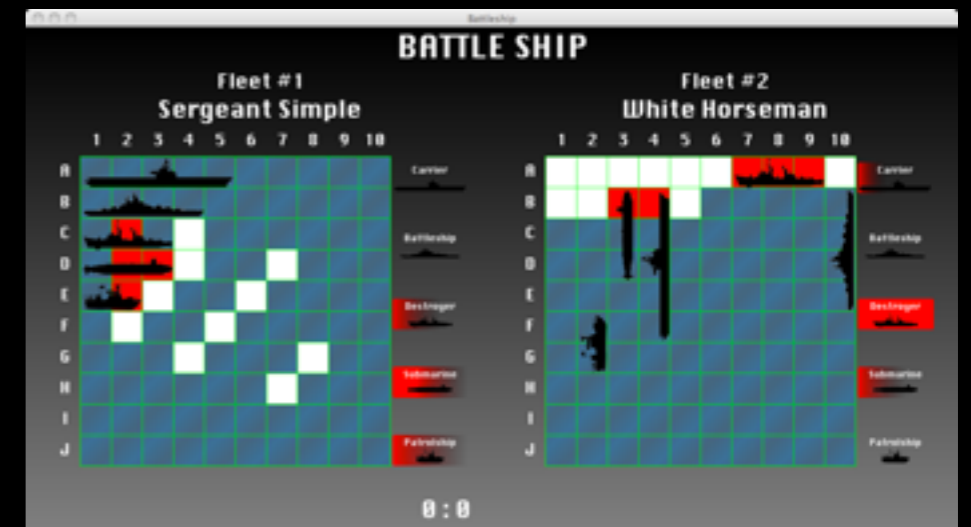
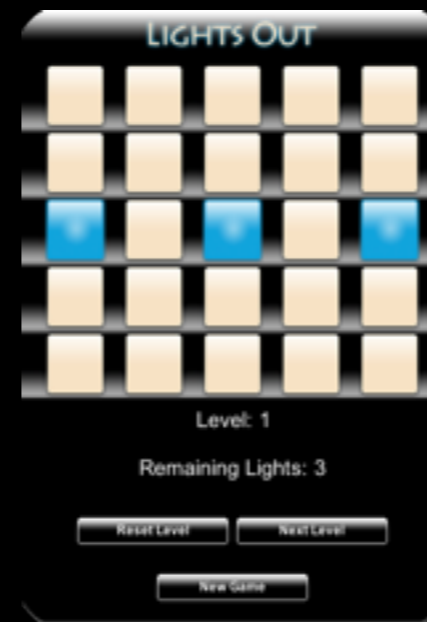
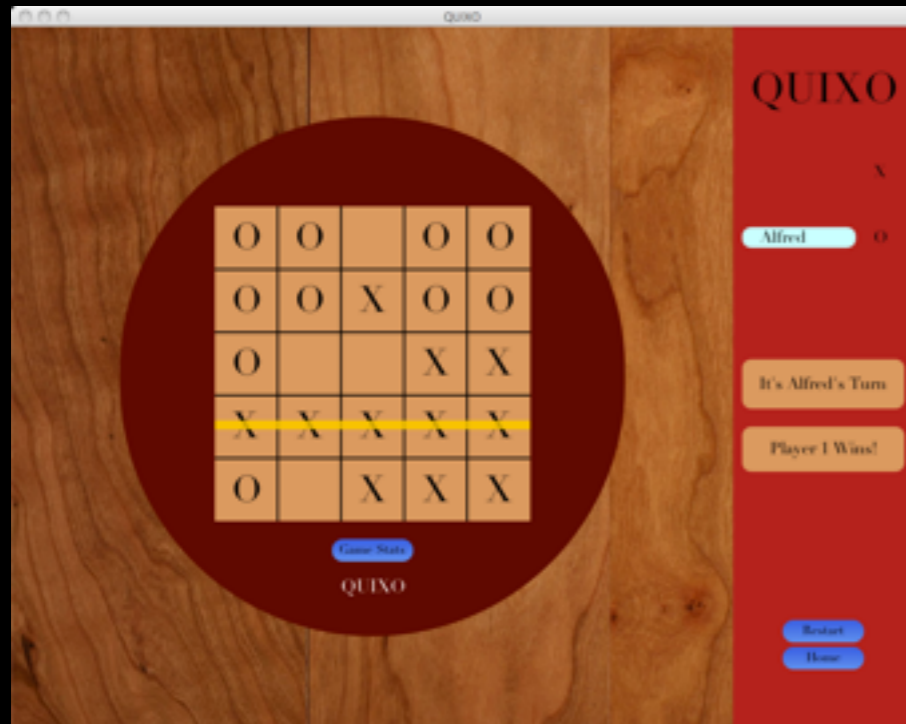


The background of the slide is a photograph of a large suspension bridge, likely the Bixby Creek Bridge, captured during the "blue hour" of twilight. The bridge's massive concrete towers and steel truss structure are silhouetted against a dark, deep blue sky. The bridge spans across a body of water, with the horizon line visible in the distance. The overall mood is serene and majestic.

Things You Can Build With Limelight

Copyright 2009 Micah Martin

Games?



Business Apps

The screenshot shows the Lighthouse Documentation website. The main content area displays the class `Lighthouse::Commands::Command`. It includes a sidebar with a class hierarchy, a search bar, and a main content area with a description, usage, and public methods.

Class Hierarchy (from sidebar):

- Class
- Lighthouse
- Animation
- Background
- Builtin
- Builtin
- CastingDirector
- Client
- Client
- Commands
- Commands
- Command
- CreateCommand
- FreezeCommand
- HelpCommand
- OpenCommand
- PackCommand
- UnpackCommand
- VersionCommand
- Data
- DSL
- DSL
- FileChooser
- FileFilter
- Gems
- Exception
- Main
- Pen
- Producer
- Production
- Prop

Command Description:

The base class for all lighthouse commands. The following is a list of them all.

Usage: `lighthouse <command> [options] [params]`

Commands:

- `create`: Creates the directories and files for a production and/or scene.
- `freeze`: freeze a gem into a production.
- `open`: Open a lighthouse production.
- `pack`: Pack a lighthouse production into a .llp file.

Attributes:

- `alias`
- `print_backtrace`: Flag. The backtrace on parse errors will be printed if true.

Public Methods:

- `description()`: Abstract class level method that returns a string description of the command. Derivatives should override this class level method.
- `install_as(name)`: Sets the alias for the derivative command and installs it in the command listing. All derivative must call this method exactly once.
- `build_options(spec)`: Abstract method to define the command line options in the OptionParser. Derivative should override this method if they offer command line options.
- `do_requires()`

The screenshot shows the Fresnel - lighthouse application. It features a navigation bar with links for 'Add Ticket', 'Add Account', 'Go to Website', and 'Logout'. Below the navigation bar is a search bar and a list of tickets.

Navigation Bar:

- Select Project: Lighthouse
- Add Ticket
- Add Account
- Go to Website
- Logout

Search:

Open Tickets: [Search]

Milestones:

- All Milestones
- Lighthouse 0.2
- Lighthouse 0.3
- Lighthouse 0.4
- Lighthouse 0.5
- Lighthouse 0.6
- Lighthouse 1.0
- Future

Tags:

- All Tags
- accessibility
- bug
- documentation
- feature_request
- release
- screenreader
- text
- widgets

Tickets Table:

Title	State	Last Activity	Assigned User
spec_helper generator	new	November 09, 2009 @ 04:00 AM	
Trouble opening lighthouse...	new	November 03, 2009 @ 11:12 PM	
Lighthouse applications ar...	new	October 30, 2009 @ 10:13 PM	
Rich Text	open	October 30, 2009 @ 01:50 AM	Micah Martin
Animated GIFs	new	October 30, 2009 @ 01:50 AM	
0 to Full size doesn't la...	new	October 30, 2009 @ 01:50 AM	
Hover Not extended	new	October 30, 2009 @ 01:50 AM	
Looped audio	new	October 30, 2009 @ 01:50 AM	
Selectable Text	new	October 30, 2009 @ 01:50 AM	
Fonts in production	new	October 30, 2009 @ 01:50 AM	
Opening a production from...	new	October 30, 2009 @ 01:50 AM	
require in styles.rb	new	October 30, 2009 @ 01:50 AM	
Windows blocky font	new	October 30, 2009 @ 01:50 AM	
Text areas don't work wit...	new	October 30, 2009 @ 01:50 AM	
Document production depen...	new	October 30, 2009 @ 01:50 AM	
Multiple text boxes highl...	new	October 30, 2009 @ 01:50 AM	
Text box after tabbing awa...	new	October 30, 2009 @ 01:50 AM	
add output to lighthouse c...	new	October 30, 2009 @ 01:50 AM	

RIA (Rich Internet Applications)

twitter™



Lighthouse

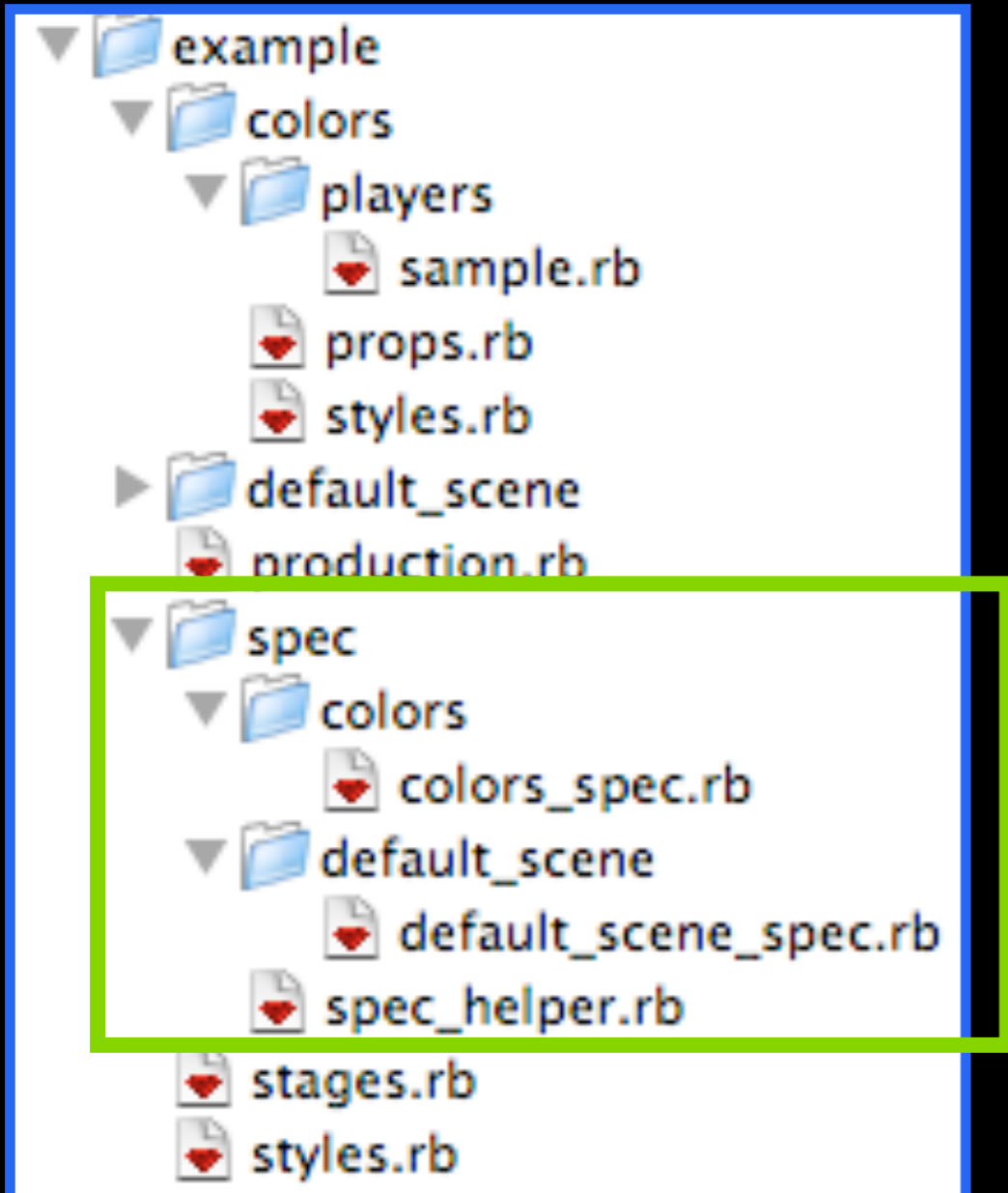


A photograph of a baby crawling on a blue safety net over a swimming pool. The net is stretched across the pool, and the baby is in the center, crawling towards the right. The background shows the pool's edge and some foliage.

Testing Limelight Productions

Copyright 2009 Micah Martin

RSpec: Unit Testing



- RSpec files generated for you
- 1 spec directory per scene by convention
- Limelight support for RSpec makes testing simple

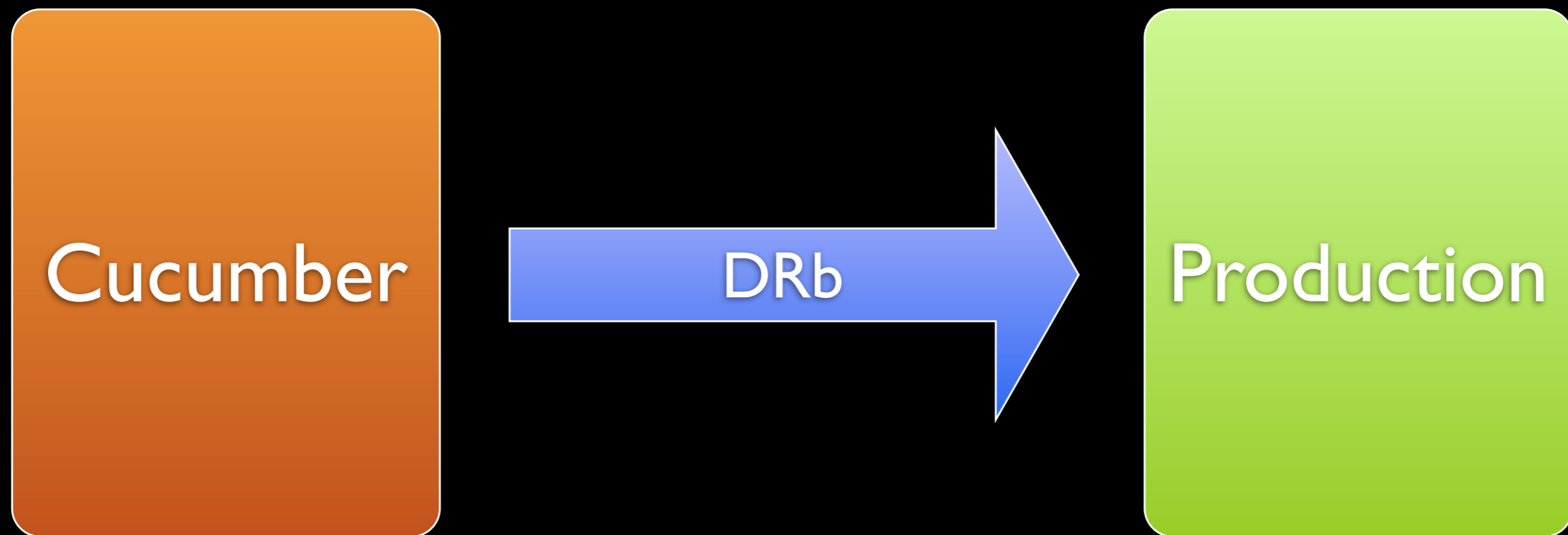
Colors Spec

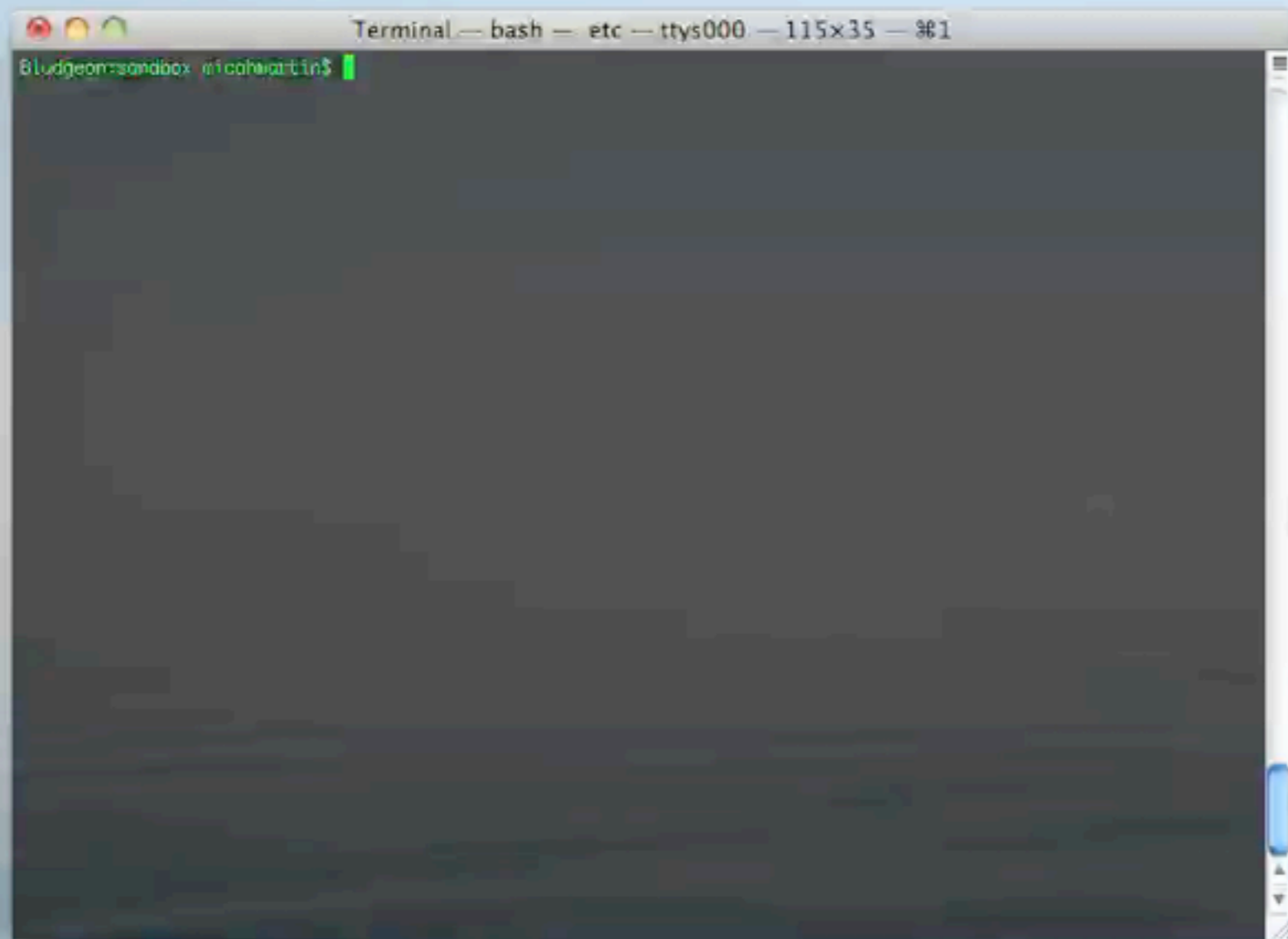
includes Limelight RSpec “goodness”

```
1 require File.expand_path(File.dirname(__FILE__) + "../spec_helper")
2
3 describe "Colors Scene" do
4
5   uses_limelight :scene => "colors"
6
7   it "should change selection color when I click on red" do
8     red_sample = scene.find_by_name("sample")[0]
9     red_sample.mouse_clicked(nil)
10    selection = scene.find("selection")
11    selection.style.background_color.should == "#ff0000ff"
12  end
13
14 end
```

provided helper method

Cucumber: Acceptance Testing





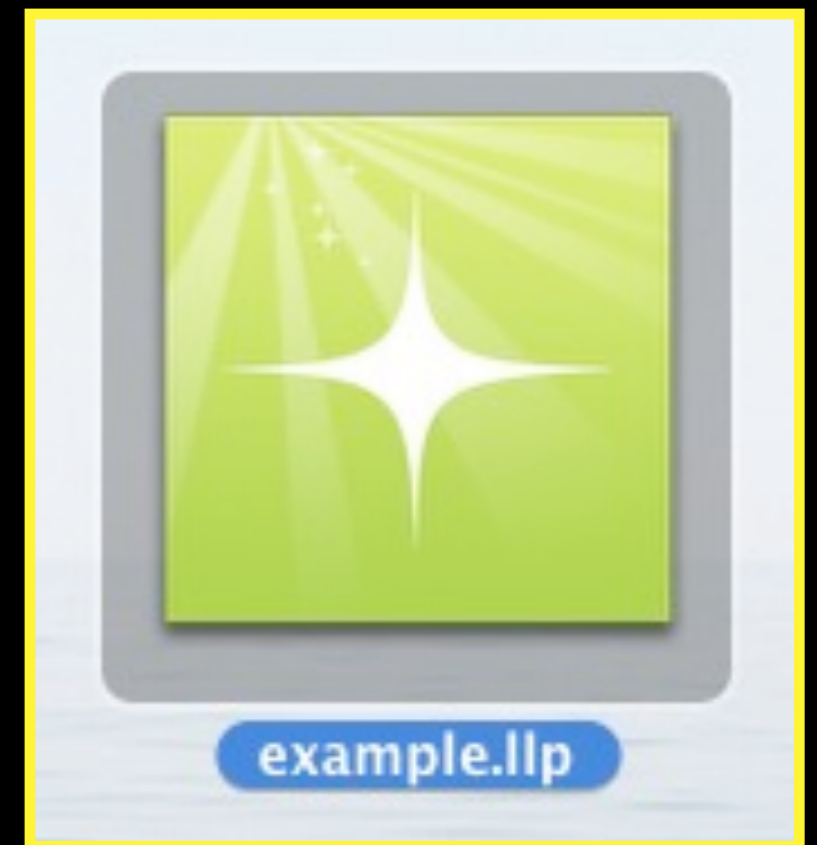
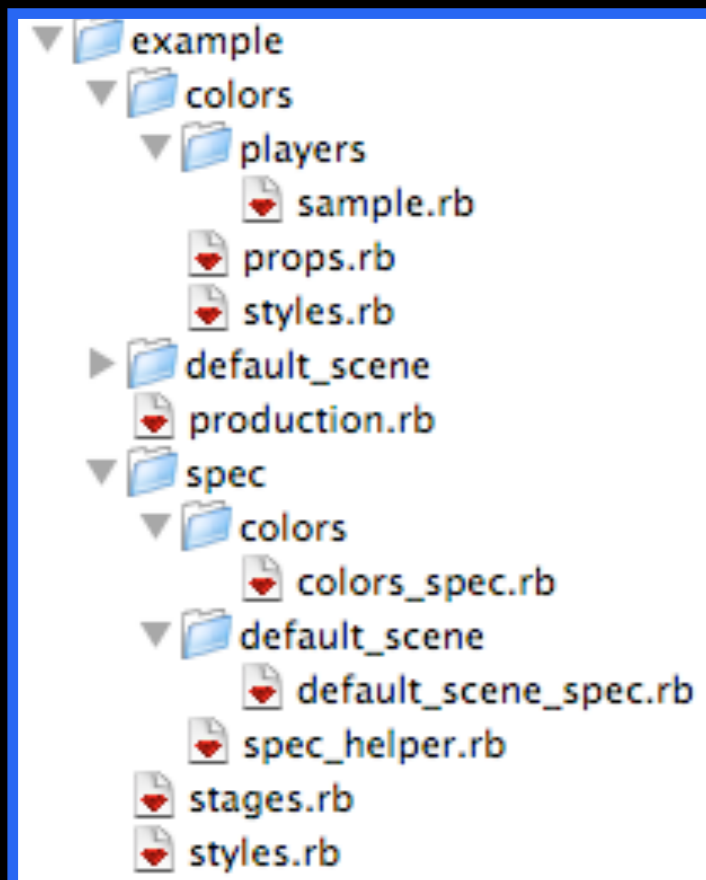


Packaging and Publishing Productions

Copyright 2009 Micah Martin

Packaging

```
1 $ jruby -S limelight pack example
```



Publishing



The Web

example.llp

1 <http://mysite.com/example.llp>

limelight playbills

Introduction

Welcome!

This is the Limelight Playbills production.

From here, you can see what productions are available on the Playbills Server.

If you see a production you like, click on the "Open it!" button to give it a whirl.

Enjoy!

Featured Productions



Limelight Docs

by 8th Light, Inc.

Open it!

Limelight Docs is a comprehensive compendium of tutorials and documentation on the Limelight framework. It's the developer's handbook for creating Limelight productions.

Available Productions



Limelight Docs

by 8th Light, Inc.

Open it!

Limelight Docs is a comprehensive compendium of tutorials and documentation on the Limelight framework. It's the developer's handbook for creating Limelight productions.



Quixo

by Matt Hinger

Open it!

Quixo is a board game invented by Thierry Chapeau and produced by Gigamic Inc. Play against your friend or the computer. The game ends when one of the players gets five in a row.



Hangman

by Micah Martin

Open it!

You know the game. Guess letters until you've figured out the word that was selected. 6 missed guesses and the hangman is toast.



Simon

by Jim Weirich and Micah Martin

Open it!

The classic game of memory. Repeat the ever growing pattern of colors. See how long you can go. This production was written in about an hour at RubyConf2008 to demonstrate rapid development with Limelight.



Langton's Ant

by Micah Martin

Open it!

Langton's Ant, invented by Chris Langton in 1986, is a simple 2D simulation demonstrating emergent behavior.



Battleship

by Micah Martin

Open it!

You sunk my battleship! This production was the arena for a software sparring competition. See if you can beat the tournament champion, the White Horseman.



Orbiter

by Robert Martin

Open it!

A simulation of gravity on orbiting masses. As masses collide, they form new, more massive bodies indicated by various colors. Quite hypnotizing!



Lights Out

by Micah Martin

Open it!

Turn 1 light off and others turn on. Try to get all the lights out. Lights Out was originally released as a hand held electronic game by Tiger Toys.

Installing Limelight



Download Installer

Run Installer



Limelight Publishing

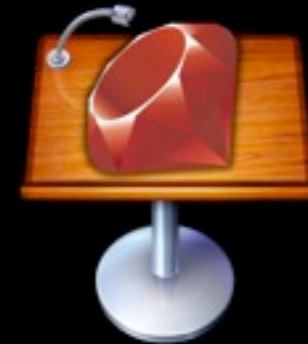
- One Install (Limelight)
- Easy Updates
- Efficient/Minimal Downloads



Finale

Limelight ...

is Ruby on the Desktop



is designed around a theater metaphor

uses a simple file-based structure



DSL

chock full of expressive DSLs

makes testing a breeze



limelight
playbills

productions are highly publishable

That's a Wrap!



limelight

<http://limelight.8thlight.com>

Micah Martin

@slagyr

8th Light, Inc.



8th light

<http://8thlight.com>

Copyright 2009 Micah Martin

That's a Wrap!

The logo for 'limelight' is presented on a green rectangular background that has a gradient from light green at the top to a darker green at the bottom. The word 'limelight' is written in a white, lowercase, sans-serif font. Above the letters 'i' and 'i' in 'light', there are small white starburst icons. The background of the logo features a sunburst effect with rays emanating from the top right corner and several small white stars scattered throughout.

limelight

<http://limelight.8thlight.com>

Micah Martin
@slagyr

8th Light, Inc.



8th light

<http://8thlight.com>

**HIRING
CRAFTSMEN**