Restructuring Clojure for a Racket-style Environment

Henry Fellows, Thomas Hagen, and Elena Machkasova

HHMI Lunch Presentation

July 1, 2015

Table of contents

Racket

- Currently use Racket
- 'toy' language
- difficult to make complex projects
- Students hitting performance issues

Why Clojure

- Used in industry (real life)
- Better on Resume
- Large community and excellent resources
- plethora of libraries (music, graphical)

What Clojure

- Clojure is a LISP
- Designed by Rich Hickey in 2007
- Functional (composition of functions)
- Built for Concurrency (simultaneous computation)

Clojure Problems

- Terrible error messages
- Missing graphics libraries for students
- Language contains misleading features
- Installing the environment is hard.

Error Messages

- Computers are dumb
- Only handle a narrow range of inputs
- Primary means of communication
- Inherently difficult to create

```
Henry:  Post office
```

Current Error Messages

- Incredibly awful
- Use strange terminology
- Meaningless to most people
- Extremely bulky

New Error Messages

- Interpret old errors
- Replace with new message
- Terminology that is friendly to novices
- Optional hints to help direct students

Recent Improvements

- Changes for Clojure 1.7.0-beta3
- Revamped hints to make them more extensible
- Made errors for infinite sequences useful `Henry: Partially print`
- Working on fixing line number reporting
- Fixed a large number of smaller issues

Future Work

- Look into integrating this with and IDE `Henry: Explain`
- Spin off our utilities into separate libraries
-

What is Quil?

- Graphical Library for Clojure
- It can:
  - Draw shapes and images
  - Move objects on the screen
  - Make games, pictures, ect..
- Quil sits on top of Clojure

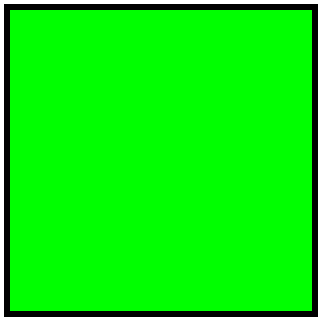
How does Quil work?

- Quil takes draw commands
- What you type: (q/rect 500 500 200 200)
- What Quil sees: Draw a rectangle at (500, 500) and make it 200 pixels wide and 200 pixels tall

Quil's fun-mode isn't enough

- Quil ONLY takes draw commands
- Quil doesn't follow MVC
- Quil code can get confusing and long

```
(q/fill 80 255 80)
(q/rect 100 100 50 50)
(q/no-fill)
(q/no-stroke)
```

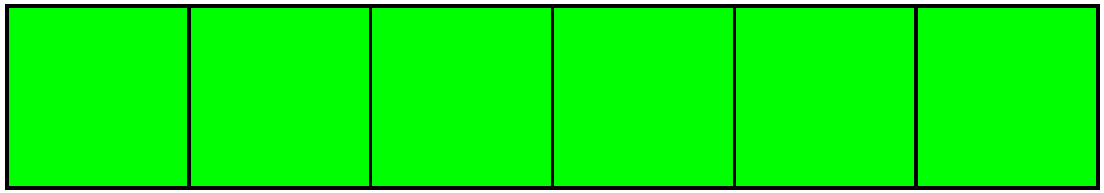


- versus

```
(def lime-rect
  (create-rect 50 50 :lime))
(ds lime-rect 100 100)
```

Six Squares

- Especially when you draw more things, such as complex shapes



Quil Code

```
  (let [x 100
   num 6
   dist (+ 100 (* (\ num 2) 50))]
(q/fill 80 255 80)
(q/rect (- dist (* 1 50)) 100 50 50)
(q/rect (- dist (* 2 50)) 100 50 50)
(q/rect (- dist (* 3 50)) 100 50 50)
(q/rect (- dist (* 4 50)) 100 50 50)
(q/rect (- dist (* 5 50)) 100 50 50)
(q/rect (- dist (* 6 50)) 100 50 50))
(q/no-fill)
(q/no-stroke)
```

Our Code

- versus

```
(def lime-rect
  (create-rect 50 50 :lime))

(def lime-rectangles
  (beside
    lime-rect lime-rect lime-rect
    lime-rect lime-rect lime-rect))

(ds lime-rectangles 100 100)
```

Our Direction

- Less paintbrush, more collage
- Create shapes, not just draw them
- Easier student code
- Give students an idea of how good software should be built

Improvements from Using Quil

- Experience with language used in the industry
- More control for future improvement
- Bigger student projects

Designing super-fun-mode

- Built on top of Quil
- Gives students functions, colors, images, ect..
- Allows for easy complex shapes

How super-fun-mode works

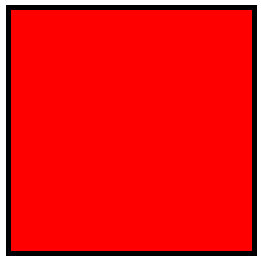
- You start by creating a shape

```
(def red-square
  (create-rect 50 50 :red))
```

- Note that creating a shape does not draw it

- From there, you can draw the shape

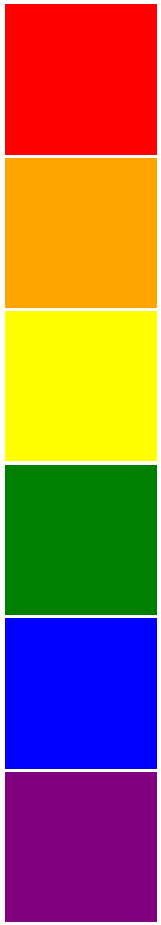
```
(ds red-square 500 500)
```



How super-fun-mode works

- You can put shapes together to make complex shapes

```
(def rainbow
     (above red-square
            orange-square
            yellow-square
            green-square
            blue-square
            violet-square))
```



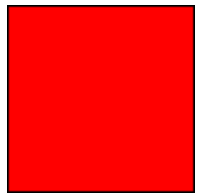
How super-fun-mode works

- You can modify the size and orientation of the shape

```
(ds (rotate-shape red-square 45)
  500 500)
```

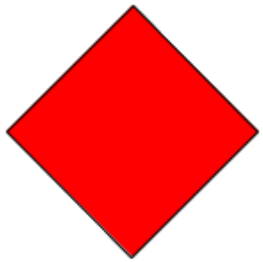
- or

```
(ds (scale-shape red-square 2 2)
  500 500)
```



- or even

```
(ds (rotate-shape
      (scale-shape red-square 2 2)
     45)
  500 500)
```



Future Work

- Fill out more functionality
  - Rotate more complex shapes
  - Pixel-detail Overlay and Overlay-Align
  - More seemless integration with Quil fun-mode

Acknowledgments

Our research was sponsored by:

- HHMI
- UMN UROP
- UMM MAP

Thank you!
Any questions?