

# Improving Clojure Usability for Introductory Course

Henry Fellows, Thomas Hagen, Sean Stockholm, Ryan McArthur,  
and Elena Machkasova  
*HHMI Lunch Presentation*  
*July 1, 2015*

# Table of contents

- 1 Introduction to the Project
- 2 Error Handling
- 3 Clojure's Graphical Library

# Goals

- Integrate Clojure into an introductory CSci class
- Currently use Racket
  - Limited teaching language
  - Difficult to make complex projects
  - Students hitting performance issues

# What is Clojure?

- Clojure is a programming language
- Built on top of the Java programming language
- Designed by Rich Hickey in 2007
- Functional (composition of functions)
- Built for concurrency (simultaneous computation)

# Why use Clojure?

- Used in industry (real life)
- Better on resume
- Many programmers enjoy using Clojure
- Large community and excellent resources
- Large number of libraries (data processing, image recognition, graphical, musical)

# Issues with Clojure

- Confusing error messages
- Missing graphics libraries for students

# Error Messages

- Computers are literal
- 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
- Consistency within error messages

# Future Work

- Look into integrating this with an 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..

fun-mode

^

Quil

^

Clojure

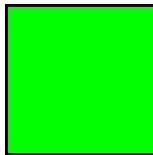
^

Java

## Quil's fun-mode isn't enough

- Quil ONLY takes draw commands
- Quil doesn't separate the model from the view
- Quil code can get confusing and long

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



# Designing super-fun-mode

- Built on top of fun-mode
- Gives students functions, colors, images, ect..
- Easy to read and change program code
- Allows for easy complex shapes

```
super-fun-mode  
^  
fun-mode  
^  
Quil  
^  
Clojure  
^  
Java
```

## 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  

```
(draw-shape red-square 500 500)
```



## super-fun-mode Complex Shapes

- You can put shapes together to make complex shapes

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



# Six Squares

- The difference becomes quite apparent with complexity





# 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

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

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

## Rotation and Scaling

- You can modify the size and orientation of the shape

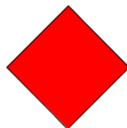
```
(rotate-shape red-square 45)
```



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



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



# Overlaying and Complex Shapes

- You can put shapes on top of each other

```
(overlay window roof)
```



```
(overlay-align :bottom :center  
  door  
  red-rect)
```



```
(scale-shape  
  (above (overlay top bottom))  
  1.4 1.4)
```



# 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

# A Few Examples

Please Enjoy a Few Live Examples

# Future Work

- Fill out more functionality
  - Rotate more complex shapes
  - Pixel-detail Overlay and Overlay-Align
  - More seamless integration with Quil fun-mode
- Open Source the project
- Integrate a Clojure sound library
- Present at Twin Cities Clojure Meetup, MN on June 8th

# Acknowledgments

Our research was sponsored by:

- HHMI
- LSAMP

Thank you!  
Any questions?