Developing a Graphical Library for a Clojure-based Introductory CS Course

Paul Schliep, Max Magnuson, Elena Machkasova

Midwest Instruction and Computing Symposium
University of Minnesota, Morris

April 25, 2014

Outline

The Project

- Contributing to ClojureEd on adapting to Clojure for an introductory course
- Objective is to develop a graphical library for Clojure
- We hope this graphical library can be useful for the introductory course
- Work in progress

Introduction to Clojure

- Developed by Rich Hickey in 2007
- Functional programming language in the Lisp family
- Runs on the JVM
- Immutable data structures and first class functions
- Data structures such as lists, vectors, hashmaps

UMM's introductory CS course

- Students are not expected to have prior programming knowledge
- The course currently utilizes Racket to help teach key concepts
- Racket is a functional language similar to Clojure
- Functional languages help students learn concepts like recursion and higher order functions
- The course makes use of Racket's graphical library

Racket graphical library game example

Benefits and limitations of Clojure

Benefits:

- Gaining traction in the industry
- Offers better parallel processing
- Integration with Java

Limitations:

- Unintuitive error messages
- Lacks a graphical library
- Lack of an IDE suitable for beginner CS students

Clojure Syntax

- Prefix notation

```
(<name of function> <argument 1> <argument 2> ...)
(+ 2 2)
-> 4
```

- Defn

```
(defn square[x] (* x x))
```

- Anonymous functions

```
(fn [x] (* x x))
```

- First class functions

```
(map square [1 2 3 4])
-> [1 4 9 16]
```

- Hashmaps

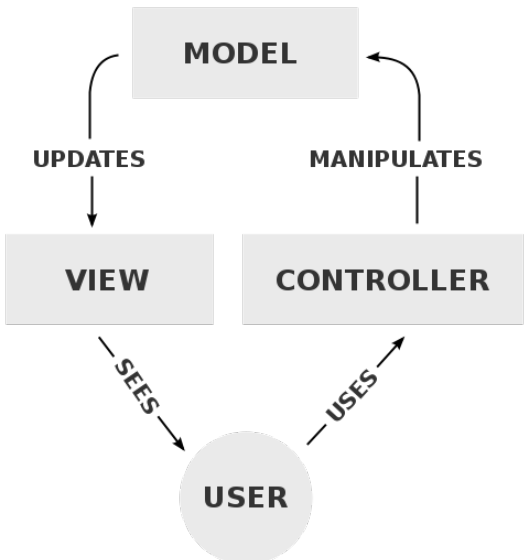
```
{:a 1 :b 2 :c 3}
```

Introduction to functional approaches

- Stylistic choice for programming
- Immutable data types
- Less dependency on order
- First class functions

Requirements for a graphical library

- Reinforce functional approaches from Clojure
- Accessible to introductory students
- Implement Model-view-controller (MVC) similar to Racket's graphical library
  - Checkers example



Overview of Quil

- Open source graphical library for Clojure
- Provides functionality suitable for introductory-level projects
- Built on top of Java Swing
- Continuously being developed

Developing programs with Quil

- Defsketch
- Works using frames and frame rate
- Draws in layers
- Supports input from keyboard and mouse

```
(defsketch example
:title "Example"
:setup setup-example
:draw draw-example
:size [400 300])
```

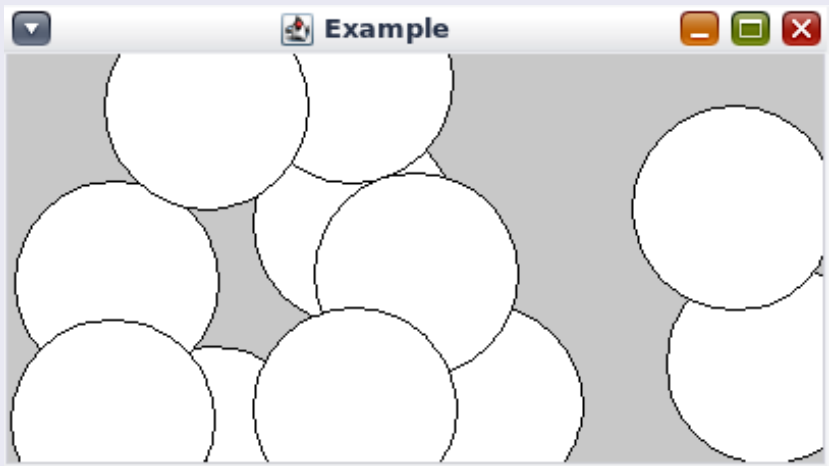
Example of a Quil program

Example Code:

```
(defn setup-example []
(frame-rate 1)
(background 200))

(defn draw-example []
(ellipse
(random (width))
(random (height))
100 100))
```

Our Image



Issues with Quil

- Imperative approaches
  - Often requires direct manipulation of state
  - Dependencies on order
  - Inconsistent with introductory course goals
- Underdocumented API

Development of the graphical library

- Abstracted over Quil's functions
  - Defsketch
  - Shapes
  - Colors
  - Text
- Handling state in a functional approach
  - Models MVC

How our graphical library works

- Separates handling of state
  - MVC
  - update
  - display

An example made using our graphical library

```
(def states
{:snake [450 450 450 470 450 490 450 510],
:snake-head [450 450],
:food [150 150],
:snake-direction "north", :score 0})

(def updates
{:setup-drawing setup
:snake update-snake
:food update-food})

(def display-order
[draw-canvas draw-food draw-snake])
```

Snake Example

Differences in handling state in Racket

Racket gives entire state to user

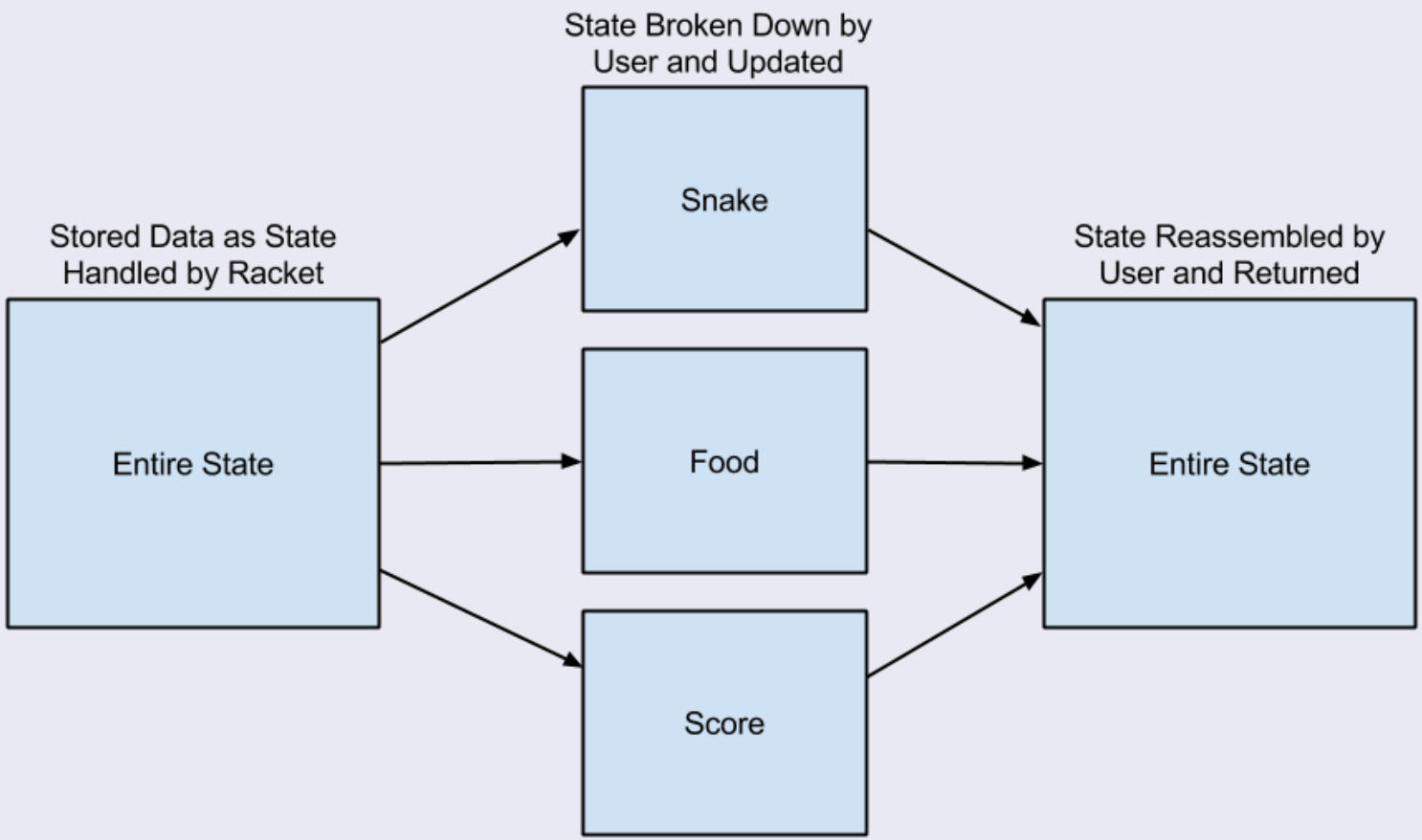
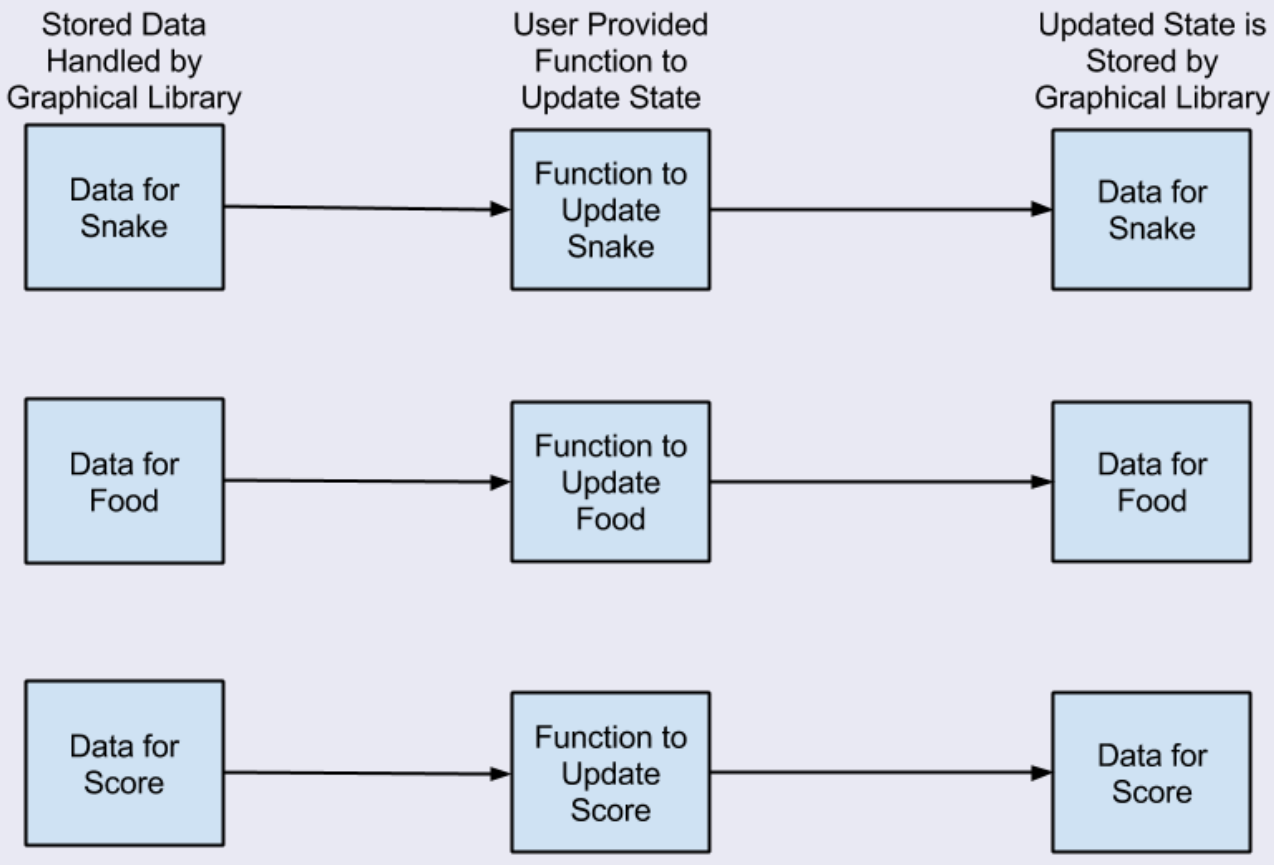


Diagram of handling state in our graphical library

Our system breaks state down for the user



Conclusions

- Good start for abstracting over Quil's functions
- More functional approach
- Graphical library shows promise

Future Work

- This is still work in progress
- Create our own macro to abstract over defsketch
- Abstract over more functions in Quil
- Develop an API with examples for students

Selected references

Selected references:

- Quil https://github.com/quil/quil
- Filleisen, M., Findler, R. B., Flatt, M. and Krishnamurthi, S. How to design programs: an introduction to programming and computing. MIT Press, Cambridge, MA, USA 2001.
- Hickey, R. The clojure programming language. In Proceedings of the 2008 symposium on Dynamic languages(New York,NY,USA,2008),DLS'08,ACM,pp.1:1-1:1.

Acknowledgements

The authors would like to thank

- Nick Skube and Niccolas Ricci
- Developers of Quil
- Friends and Family