Generative Art with Racket

Sixth RacketCon - Rodrigo Setti
Agenda

1. Demo
2. System Design
3. My Experience
demo
#lang s-exp stamps/lang

(define-shape sierp
  (triangle)
  (sierp [translate 0 0.288]
    [scale 0.5 ]
    [brightness 0.1 ])
  (sierp [translate -0.25 -0.144]
    [scale 0.5 ]
    [brightness 0.1 ])
  (sierp [translate 0.25 -0.144]
    [scale 0.5 ]
    [brightness 0.1 ]))

(start-shape sierp)
#lang s-exp stamps/lang

(define-shape tree
  (branch)
  (branch [flip 90]))

(define-shape branch
  [98 =>
    (circle)
    (circle [scale 0.9]
      [brightness 1])
    (branch [y 0.2]
      [scale 0.99]
      [rotate 3])]
  [2 =>
    (circle)
    (circle [scale 0.9]
      [brightness 1])
    (branch [y 0.2]
      [scale 0.99]
      [flip 90])
    (branch [y 0.2]
      [scale 0.6]
      [brightness 0.2]))]

(start-shape tree)
(define-shape S
  (square)
  (square [s 0.5 ] [b 1 ])
  (S [r .2 ]
    [t .7 .7]
    [s .995 ]
    [b .002 ])))

(background '(0 0 .5))

(start-shape S)
#lang s-exp stamps/lang

(define-shape start
  ((stem 8) [h 300]))

(define-shape (stem branches)
  ((loop ([i branches])
        ((branch (random-real -6 6)) [x .1]
         [r (* i (/ branches 360))] [s .7]))))

(define-shape (branch turn)
  [1 => (circle [sat .4])
    ((branch turn) [x .1]
     [s .99]
     [r turn]
     [b .01]])
  [.1 => ((branch (random-real -6 6)))]
  [.04 => ((stem (random-integer 1 4)))
     [.001 => ]]

(background '(0 0 0))
(maximum-render-cycles 100000)
(start-shape start)
System Design

- language
- macros
- shape combinators
- shape “core”

```scheme
#%module-begin

define-shape

join : map[probability, shape] → shape

shape : adjustment → renderer

renderer : device-context → list[renderer]
```
My Experience

**What is Great**

- DrRacket
- Documentation
- Packages
- Macro and language system
- Typed racket
- Profiling

**Areas for Improvement**

- Untyped matrix performance
thank you

github.com/rodrigosetti/stamps