ScriptingLanguageSpecification

Matthew Bregg

December 6, 2015

Contents

1 Scripting Language Specifications/Overview

For the scripting language, we have choosen a lisp style language. While far from a complete lisp, it has a syntax similar to scheme, and could easily be extended, without breaking existing scripts.

1.1 Built in functions

1.1.1 define

```
(define name value)
(define fooConstant 3)
(define (sqr x) (* x x))
```

Assigns to the name the given value. This version does support defining functions atm. (See `http://www.scheme.com/tspl2d/binding.html`)

- Right now, defines may be nested.
- A variable name can't start with a number, to make it determinable from an int atom.
  - It must start with an alpha character.

1.1.2 Basic number ops

Basic number ops, including +,-,/,*,^. Takes in two args, returns a third with the value.

```
(+ 1 2)
(- 1 2)
(* 1 2)
(/ 1 2)
(^ 1 2)
```

1.1.3 concat-string

Takes in n string, or numbers, returns one string of all those concattenated

```
(concat-string "foo" 3 "bar")
(concat-string "Foo" "bar")
```

1.1.4 cons

Takes two arguments, returns a pair holding the two arguments as one object

```
(cons a b)
(cons 1 2)
```

(See `http://download.plt-scheme.org/doc/4.2.4/html/guide/Pairs__Lists__and_Scheme_Syntax.html`)

1.1.5 card

Takes a card-size, name, and two layouts, one for the front, one for the back, and then the shape of the card.

```
(card card-size "name" frontLayout backLayout shape)
```

When a card is rendered, it will pair the layout with the wholeCard position-scaled (0 0 100 100)

1.1.6 If statements

The language has basic support for if statements. The language takes in three s expressions, the first is evaluated, if it is "#f", then the third token is evaluated, and the result returned. If it is anything else, then the second token is evaluated, and the result returned.

```
(define x...
(define y...
(if "#f" 3 4)
(if (== x y) (layout "foo") (layout "bar"))
```

1. Logic Operators The language supports two logic operators, `=, and !`. These return "#f", or "#t".

   ```
   (!= "foo" "bar")
   ```

2. NOTE Using "#f" and "#t" rather than #t #f, small variation from normal lisp.

1.1.7 render

Takes in a single card, a list of cards, or a pair of cards, and 0-n families. Renders them.

```
(render card family)
(render cards family)
(render (cons carda cardb) family)
(render some-cards family0 ... familyn-1)
```

Note: The family is an optional argument, leaving it empty is the same as calling

```
(render cards (family))
```

Which runs render with an empty family.

1.1.8 list

Takes in a n arguments, and returns a list of them.

```
(list N0 ... Nn-1)
(list 1 2 3 4 5 6 7)
```

1.1.9 position-scaled

Takes in a x-offset%, y-offset%, and a scale-width% and scale-height%, and returns a position-scaled object.

```
(position-scaled x-offset% y-offset% scale-width% scale-height%)
(position-scaled 0 0 50 50)
(define wholeCard (position-scaled 0 0 100 100))
```

1.1.10 leaf-layout

Returns a layout. Takes in a decal, or a string. In the event a string is given, the decal will be looked up in the family. This layout can then be used with the above layout function.

```
(layout image)
(layout foobarImage)
(layout "foo")
(layout (color-decal "white"))
```

1. Leaf-Layout options A Leaf-Layout can be given a third argument, to determine some extra behavior. Takes an extra parameter, either a W, or an H, A, or O.

   - If W, width will be at most maximimum width of an image.
   - If H, height will be at most, maximum height of given image
   - If A, the original aspect ratio will be maintained.
   - If O, original size will be mantained.
   - IF S, stretch to fit.
   - All the options aside S, which does need to, will add transparent padding to return a size render desires.
   - The default is "S", so calling with the "S" argument is the same as not having a third argument

1.1.11 layout

Creates a Layout object. A layout contains 0-n tuples of layouts position-scaleds, and shapes. Takes 0-n tuples of layouts position-scaleds and shapes as arguments.

```
(Layout
  (list layout0 position-scaled0 shape0)
  (list layout1 position-scaled1 shape1)
  ...
  (list layoutn-1 position-scaledn-1 shapen-1))

(Layout
  (list layoutFoo position-scaledFoo rectanglebar)
  (list (layout foobarImage) wholeCard rectanglefoo)
  (list (layout "foo") (circle 3.14))
)
```

1.1.12 family

Creates a map of strings to decals, a family. Takes in a name, and N pairs.

- Requires a family name.
  - The family name is added to the card name when a card is rendered, to avoid name collisions when rendering the same card with multiple families.

    - If a multiple families given, append the names of all the families.

```
(family name pair0 ... pairn-1)
(family "fooFamily" (cons "foo" fooImage) (cons "bar" barImage))
```

1.1.13 eval-file

Takes in n filepaths, evals each file in given order

```
(eval-file "filename.filename")
(eval-file "foo.script")
(eval-file "foo.script" "bar.script")
```

Evals foo.script. Returns null.

1.1.14 Decals

- Image Decal

```
(image "filepath.[jpg|png|etc]")
(image "foo.jpg")
```

- Color Decal
    - A decal takes in a color

      ```
      (color-decal "color")
      (color-decal "white")
      ```

- String decal

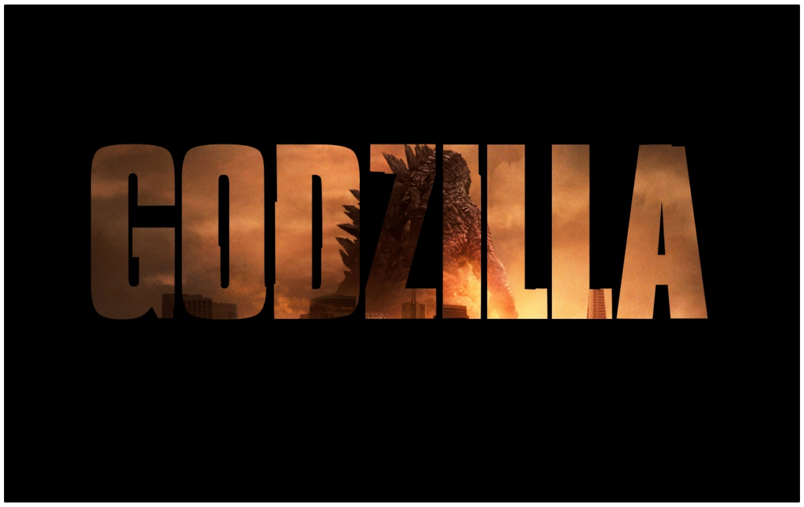
A string from a given font.

```
(string "StringText" "Font" "Color" Size)
(string "Hello World!" "Arial.font" "Red" 12)
(string "1" "Arial.font" "Red" 12)
```

- Mask Decal
    - Takes a decal, foo, and has the non transparent portions of foo replaced with corresponding portions of bar. What portion of bar maps to what portion of foo is determined by the position scaled.

```
(define foo-decal (image "foo.png"))
(define bar-decal (image "bar.png"))
(mask-decal foo-decal bar-decal (position-scaled 0 0 100 100))
```

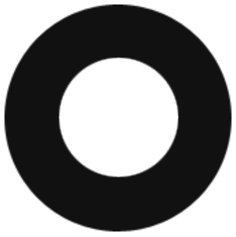
  - Example of effect, mask would leave transparent back ground.



- Inverted Mask Decal
  - Takes a decal, foo, and a decal bar, and has the non transparent parts of bar removed from foo, leaving a bar shaped hole in foo.
  - The position is used to determine where bar should be cut from foo.

```
(define foo-decal (image "foo.png"))
(define bar-decal (image "bar.png"))
(inverted-mask-decal foo-decal bar-decal (position-scaled 0 0 100 100))
```

  - Example: A circle hollowed out



- Rotate Decal

  - Takes in a decal, and a number, and returns a rotated version of that decal

    ```
    (define foo-decal (image "foobar.png"))
    (rotate-decal foo-decal 90)
    ;;Returns a decal rotates 90 degrees.
    ```

- Corner Rounding Decal

  - Takes a decal, and rounds the corners.

    ```
    (define foo-decal (image "foobar.png"))
    (corner-rounder foo-decal)
    ;;Returns a decal with its corners rounded.
    ```

- Crop Decal

  - Takes in a decal, and a position-scaled, crops the decal to the area the position-scaled defines.

```
(define foo-decal (image "foobar.png"))
(crop-decal foo-decal (position-scaled 50 50 50 50))
;;Returns a decal cropped to the middle.
```

1.1.15 Shapes

- Rectangle

```
(rectangle width height)
(rectangle 100 200)
```

- Triangle

```
(triangle lengthA lengthB lengthC)
(triangle  100 200 300)
```

- AnyShape

  - Connect point0 -> point1, and then pointn-1 -> point0 to make a shape

```
(any-shape point0x point0y point1x point1y ... pointn-1x pointn-1y)
(any-shape  100 100 200 200 300 300)
```

- Circle

```
(circle radius)
(Circle 100)
```

1.1.16 Size

A size is used by a card to determine how many pixels it will be.

```
(size width height)
```

2 Config file

- Allows one to set various options
- Current options are
  - script-file
    - Specify the script to run
    - No default, can be overridden by terminal args
  - output-format
    - Specify what format to output in
    - Defaults to png
  - output-file
    - Specify where to output the result to
    - Defaults to ./
  - logfile
    - Specify where to log to
    - Defaults to .cardlog
  - load-builder
    - Takes in a name, and the path to a builder java file.
    - Loads said builder into script evaluator
    - This defines a new keyword in the language, and serves as as plugin.

      ```
      (set-option "load-builder" "name" "path")
      ```
- Each option is enter in this format

```
(set-option "option-name" values)
```

-So for example

```
(set-option "output-dir" "./")
```

2.1 Standard Library

The program will ship with numerous built in functions. These functions will have std- prepended to their name, and will provide various functionalities that preimplemented in the language, to save the user the hassle of implementing them. All these functions could be done normally, in the scripting language.

This will be implemented in a std-functions.script, which will the driver will run through the evaluator when the program first runs.

2.1.1 add-border

1. Parameters
    - input-layout : The layout to add a border to
    - color : The color to make the border
    - size : how thick to make the border
2. Return value
    - A layout with the border added
3. Example implementation

```
(define (std-add-border input-layout color size)
  (layout
   (list input-layout
         (position-scaled size size
                          (- 100 size) (- 100 size)))
   (list
    (leaf-layout (color-decal color)) whole-card)
   )
  )
```

2.1.2 get-colored-shape-in-a-layout

1. Parameters
    - shape : The shape to put in the layout
    - color : the color to make it.
2. Return value

- A layout with a leaf, which is paired with the shape, holding the color, over the whole layout.

3. Example implementaiton

```
(define (std-get-colored-shape-in-a-layout shape color)
  (layout
   (list
    (leaf-layout
     (color-decal color)) whole-layout shape)
   )
  )
```

2.1.3 get-hollow-decal

1. parameters
   - decal : The decal to be hollowed
   - size : A number, determines how much to hollow
2. Return value
   - A decal that has been hollowed
3. Example implementation

```
(define (std-get-hollow-decal decal size)
  (inverted-mask-decal decal decal
                       (position-scaled size size
                                        (- 100 size) (- 100 size)))
  )
```

2.1.4 Some constants

- std-whole-layout : (position-scaled 0 0 100 100)
- std-pi : Value of pi

2.1.5 Many more things!