

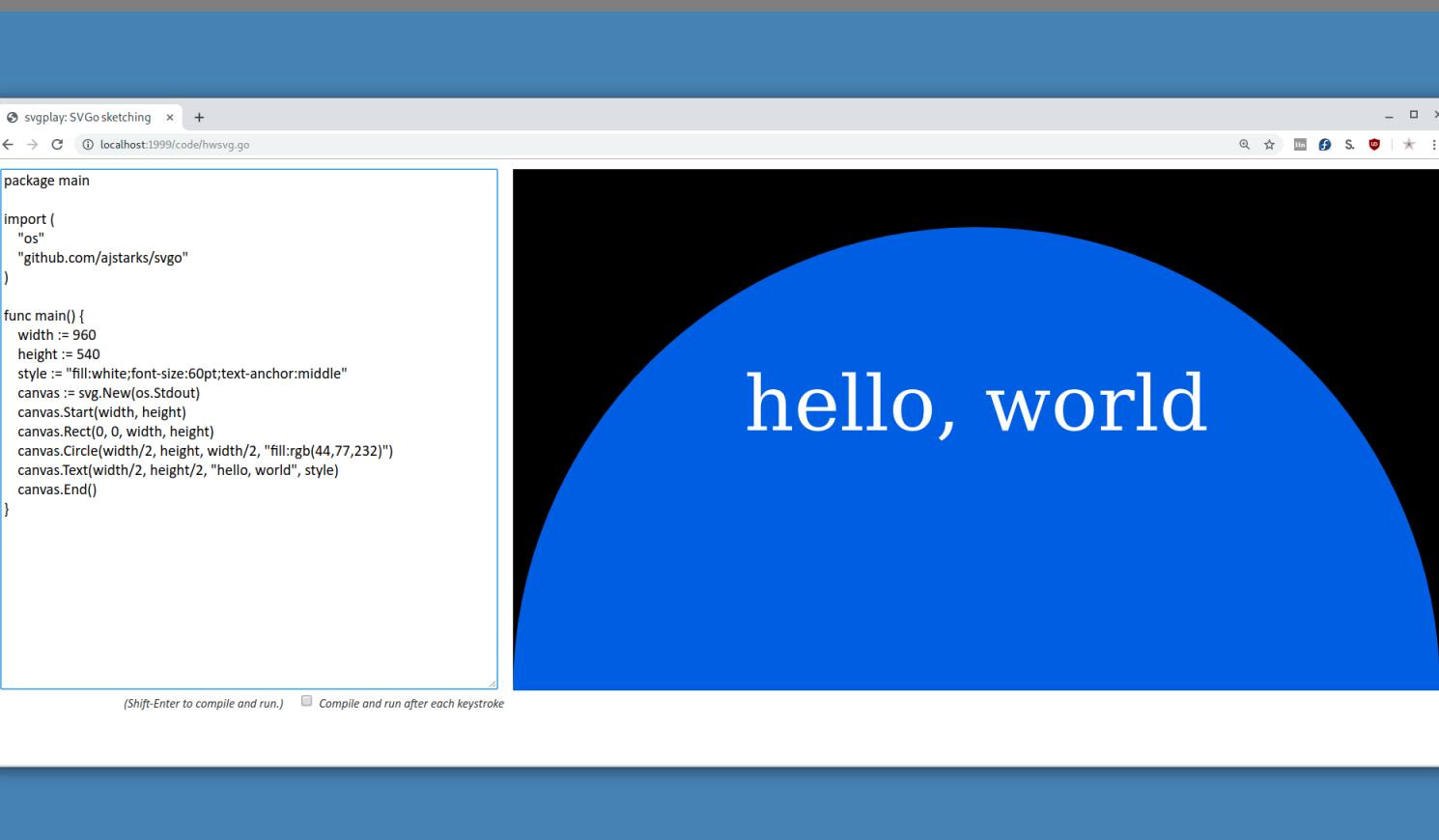
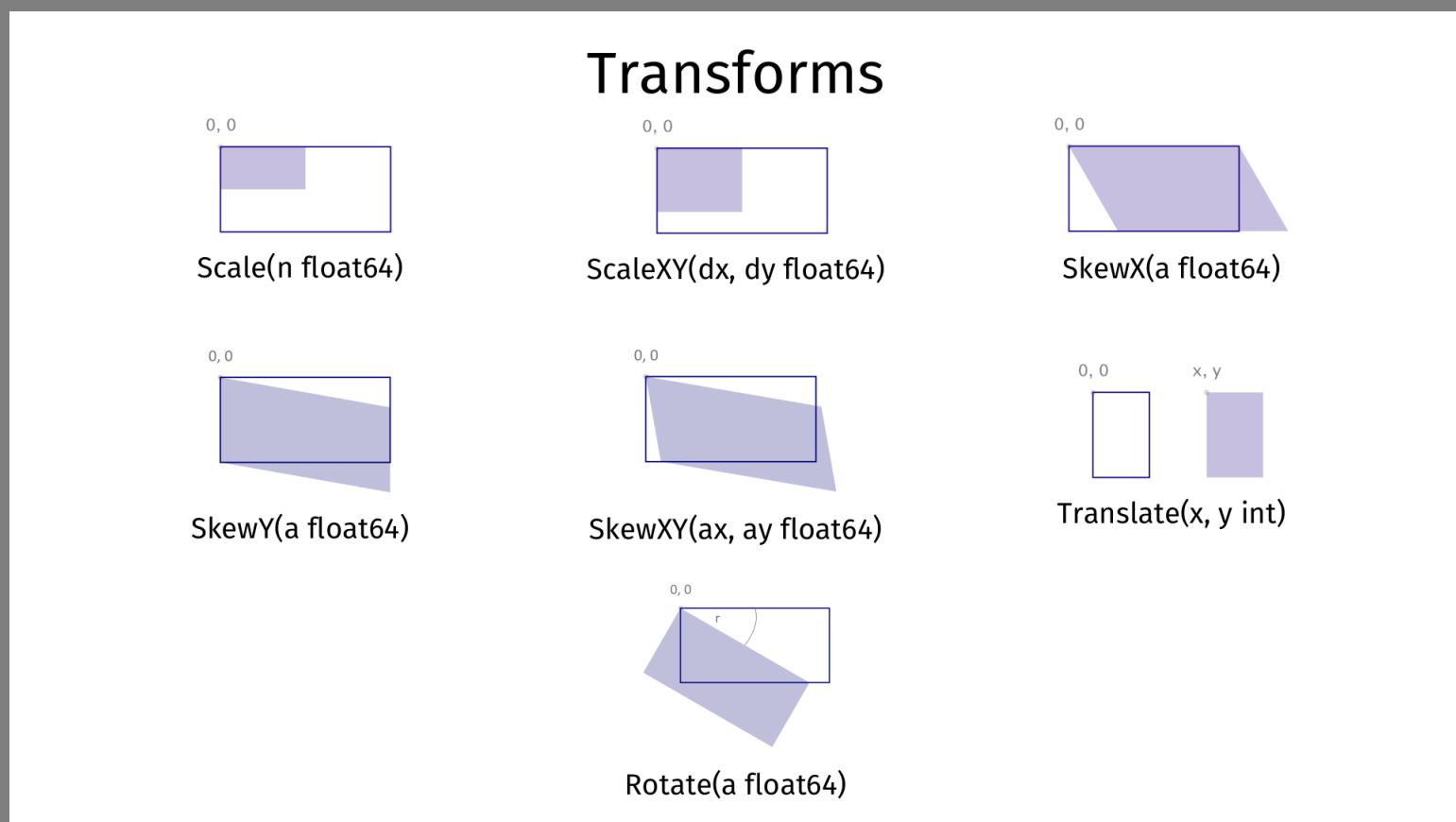


SVG

# Workshop

Anthony Starks  
@ajstarks

# You will learn:



SVGo API

Sketching with code

SVGo with other APIs

# You will need:

```
ajstarks@slab:~$ cd  
$ go version  
go version go1.12.9 linux/amd64  
$ go get github.com/ajstarks/svgo/...  
$ git clone https://github.com/ajstarks/svgo-workshop  
Cloning into 'svgo-workshop'...  
remote: Enumerating objects: 17, done.  
remote: Counting objects: 100% (17/17), done.  
remote: Compressing objects: 100% (14/14), done.  
remote: Total 531 (delta 5), reused 14 (delta 3), pack-reused 514  
Receiving objects: 100% (531/531), 75.70 MiB | 33.60 MiB/s, done.  
Resolving deltas: 100% (111/111), done.  
$ cd svgo-workshop/code/svgplay-samples/  
/home/ajstarks/svgo-workshop/code/svgplay-samples  
$ svgplay  
2019/08/18 21:52:37 💀💀💀 Warning: this server allows a client connecting to 127.  
0.0.1:1999 to execute code on this computer 💀💀💀
```

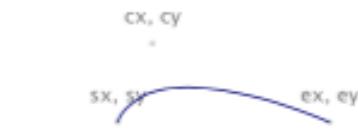
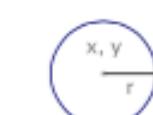
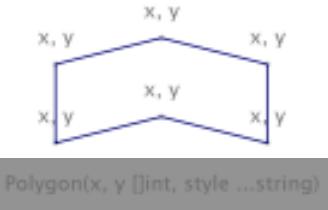
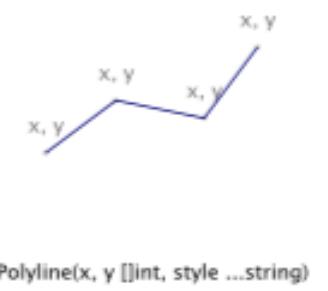
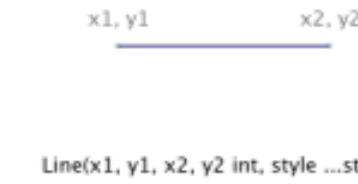
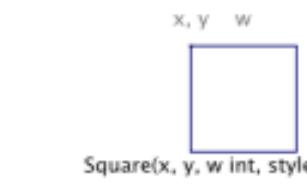
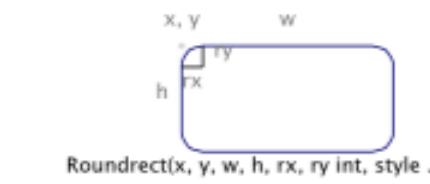
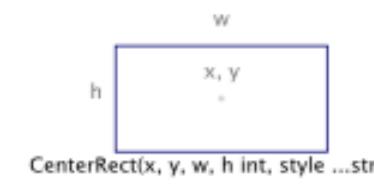
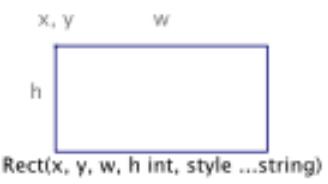


Go installation and command line

Modern Browser

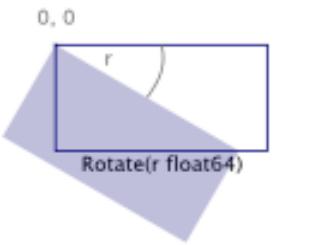
# SVG Go Library

github.com/ajstarks/svggo



\$ go get github.com/ajstarks/svggo/...

\$ git clone https://github.com/ajstarks/svgo-workshop



It's "holy" & "dandy" to draw text  
duotone

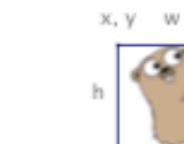
hello, this is SVG

Text(x, y int, s string, style ...string)

Textpath(s, pathid string, style ...string)



Path(s string, style ...string)



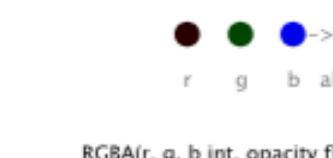
Image(x, y, w, h, int path string, style ...string)



Grid(x, y, w, h, n int, style ...string)



RGB(r, g, b int)



RGBA(r, g, b int, opacity float64)

New(w io Writer)  
Start(w, h int, options ...string)/End()  
StartView(w, h, minx, miny, vx, vy int)  
Groups(...string)/End()  
Gstyle(s string)/End()  
Gtransfrom(s string)/End()  
Gid(id string)/End()  
ClipPath(s ...string)/ClipEnd()  
Defn()/DefEnd()  
Marker() /MarkerEnd()  
Pattern() /PatternEnd()  
Desc(s string)  
Title(s string)  
Script(type, data ...string)  
Mask(id string, x,y,w,h int, style ...string)/MaskEnd()  
Link(href string, title string)/LinkEnd()  
Use(x int, y int, link string, style ...string)

specify destination  
begin/end the document  
begin/end the document with viewport  
begin/end group with attributes  
begin/end group style  
begin/end group transform  
begin/end group id  
begin/end clip path  
begin/end a definition block  
begin/end markers  
begin/end pattern  
set the description element  
set the title element  
define a script  
begin/end mask element  
begin/end link to href, with a title  
use defined objects

Textlines(x, y int, s []string, size, spacing int, fill, align string)



Scalable Vector Graphics

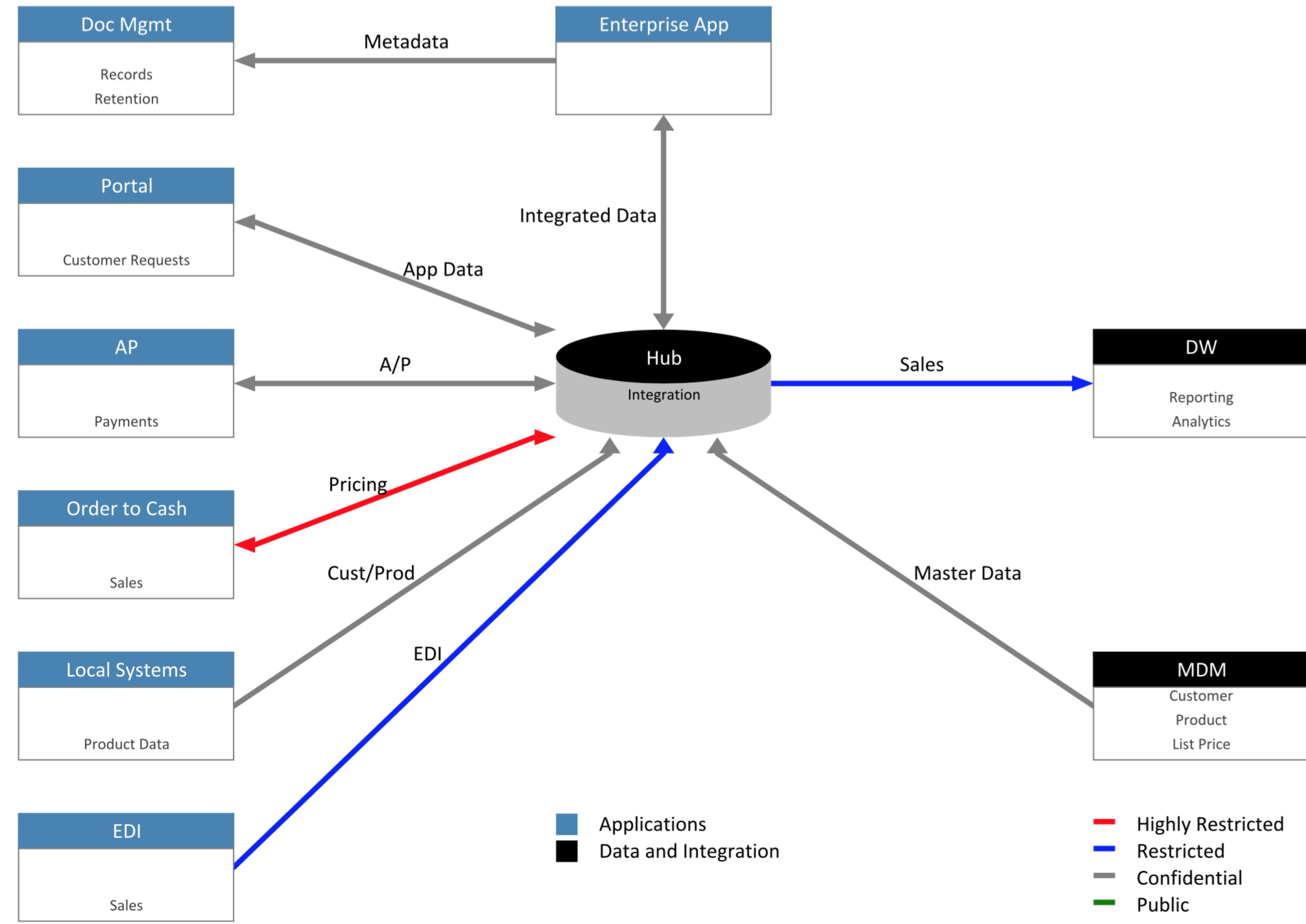


io.Writer

# SVGo Clients in the Repo

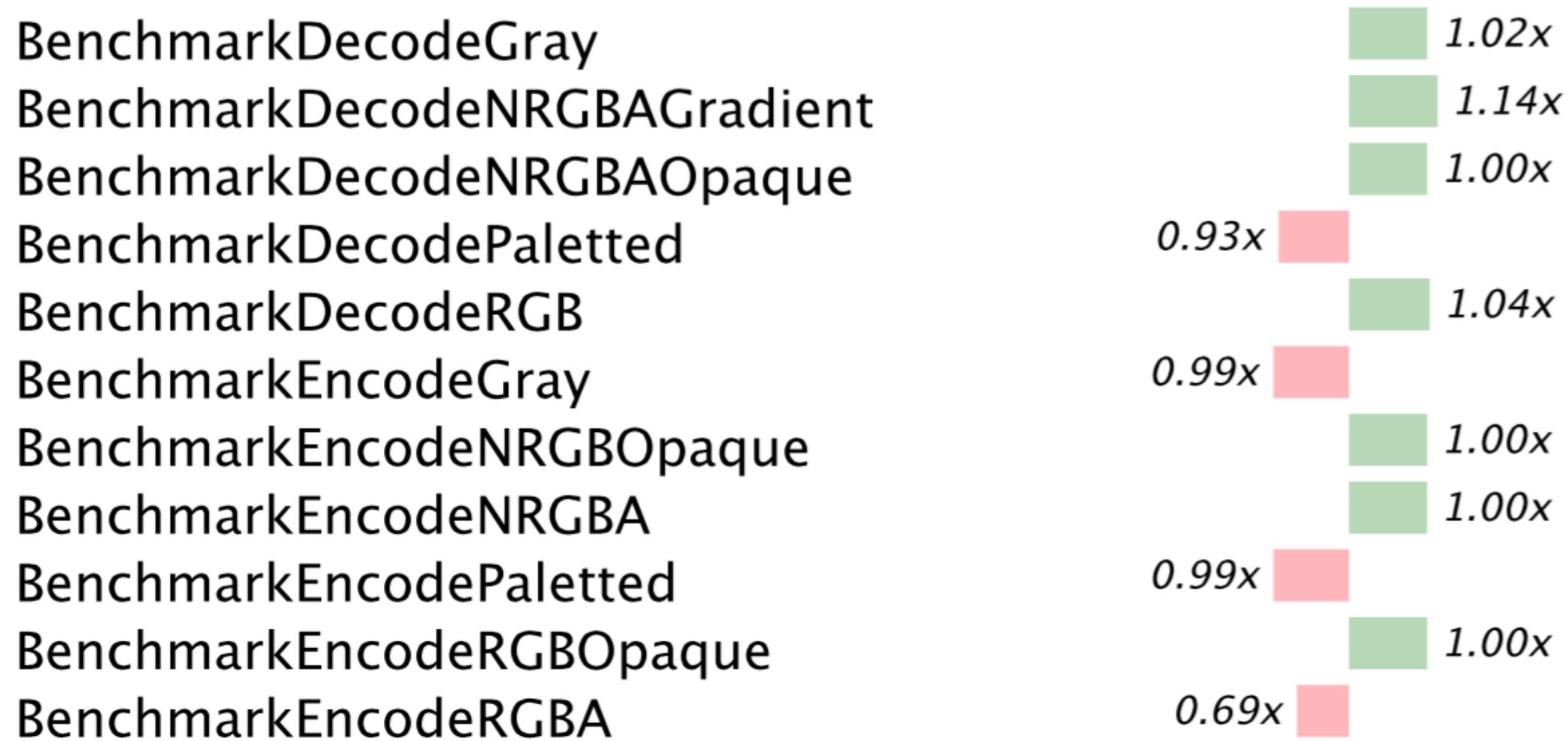
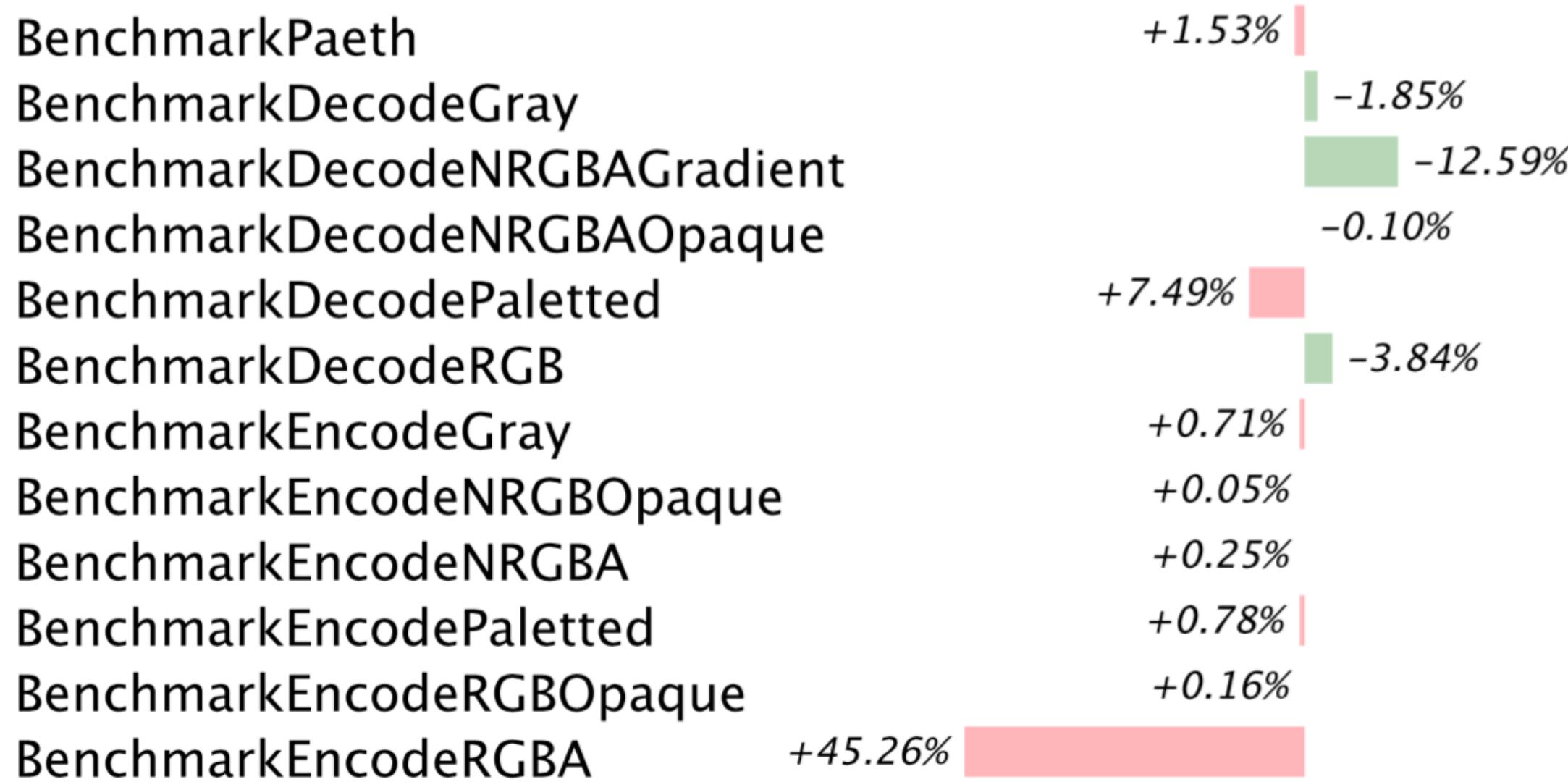
svgdef	Creates a SVG representation of the API	pattern	Test patterns
android	The Android logo	planets	Show the scale of the Solar system
bubtrail	Bubble trails	pmap	Proportion maps
bulletgraph	Bullet Graphs (via Stephen Few)	randcomp	Compare random number generators
colortab	Display SVG named colors with RGB values	richter	Gerhard Richter's 256 colors
compx	Component diagrams	rl	Random lines
flower	Random flowers	skewabc	Skew ABC
fontcompare	Compare two fonts	stockproduct	Visualize product and stock prices
f50	Get 50 photos from Flickr from a query	svgopher	SVGo Mascot
fe	Filter effects	svgplay	SVGo sketching server
funnel	Funnel from transparent circles	svgplot	Plot data
gradient	Linear and radial gradients	svgrid	Compose SVG files in a grid
html5logo	HTML5 logo with draggable elements	tsg	Twitter Search Grid
imfade	Show image fading	tumblrgrid	Tumblr picture grid
lewitt	Version of Sol Lewitt's Wall Drawing 91	turbulence	Turbulence filter effect
ltr	Layer Tennis Remixes	vismem	Visualize data from files
marker	Test markers	webfonts	"Hello, World" with Google Web Fonts
paths	Demonstrate SVG paths	websvg	Generate SVG as a web server

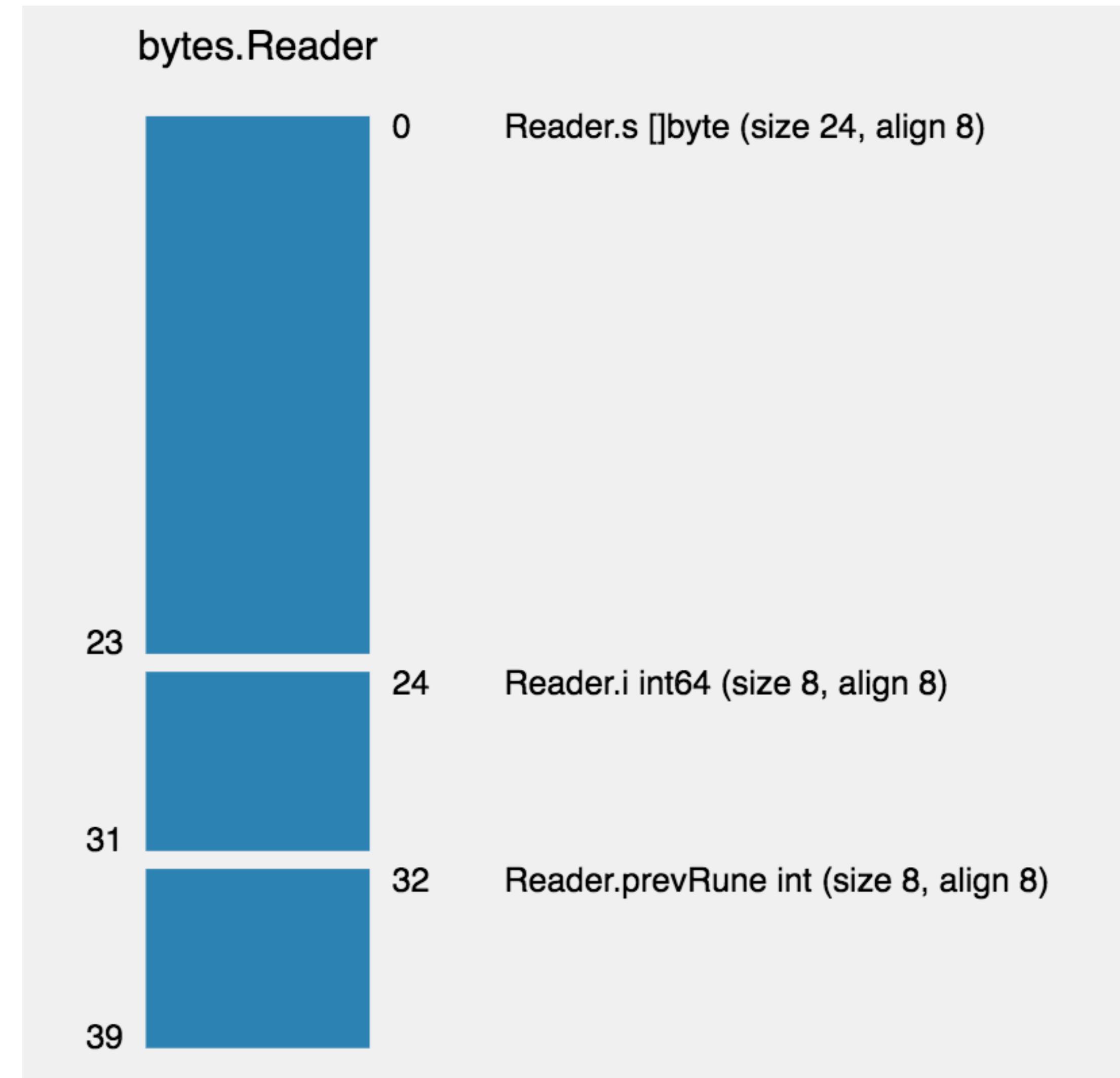
# SVGo Clients



# benchpng.txt

*image/png package: Go 1.3 vs Go 1.4*





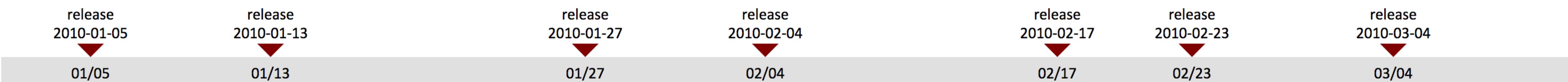
```
structlayout -json bytes Reader | structlayout-svg -t bytes.Reader > reader.svg
```

# Go Programming Language Release History

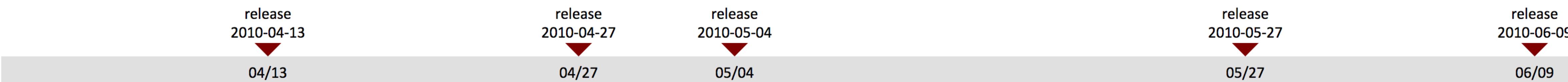
4Q 2009



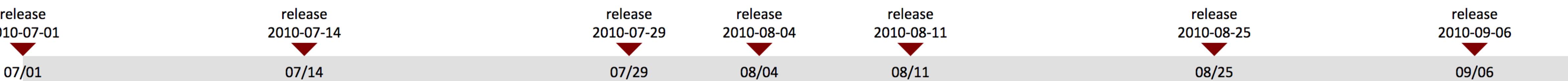
1Q 2010



2Q 2010

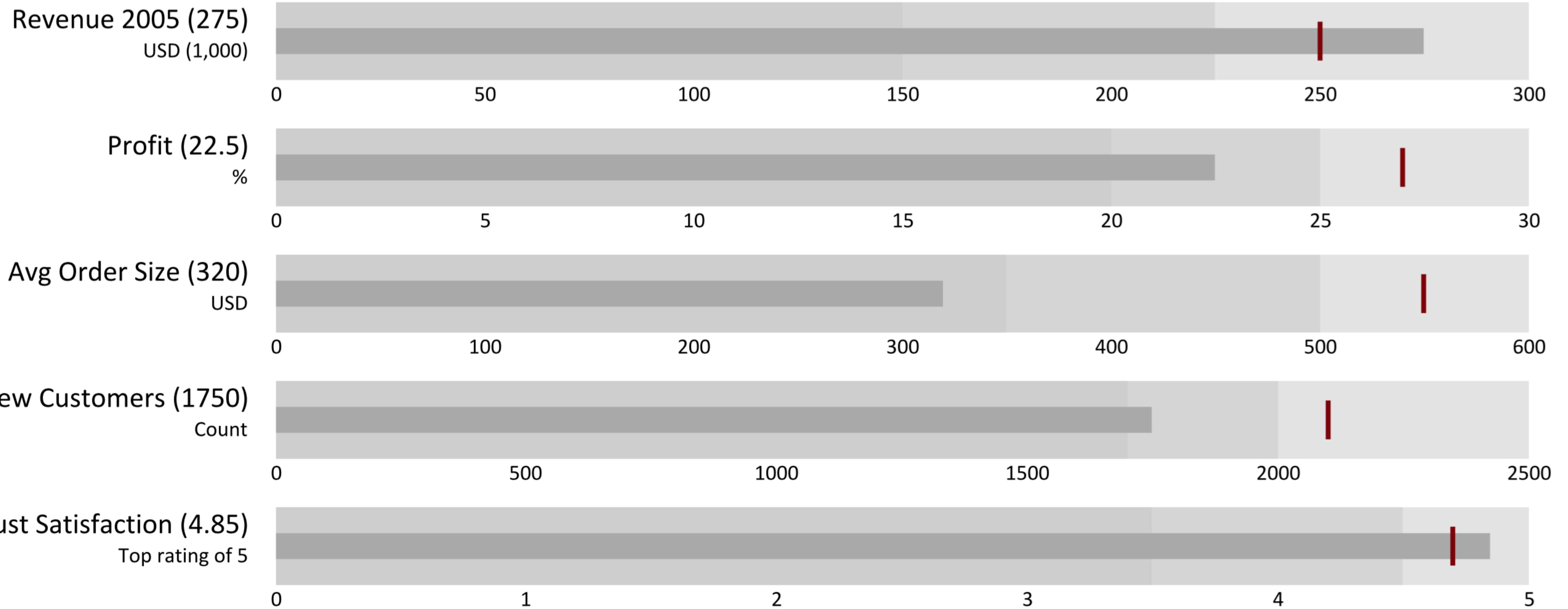


3Q 2010



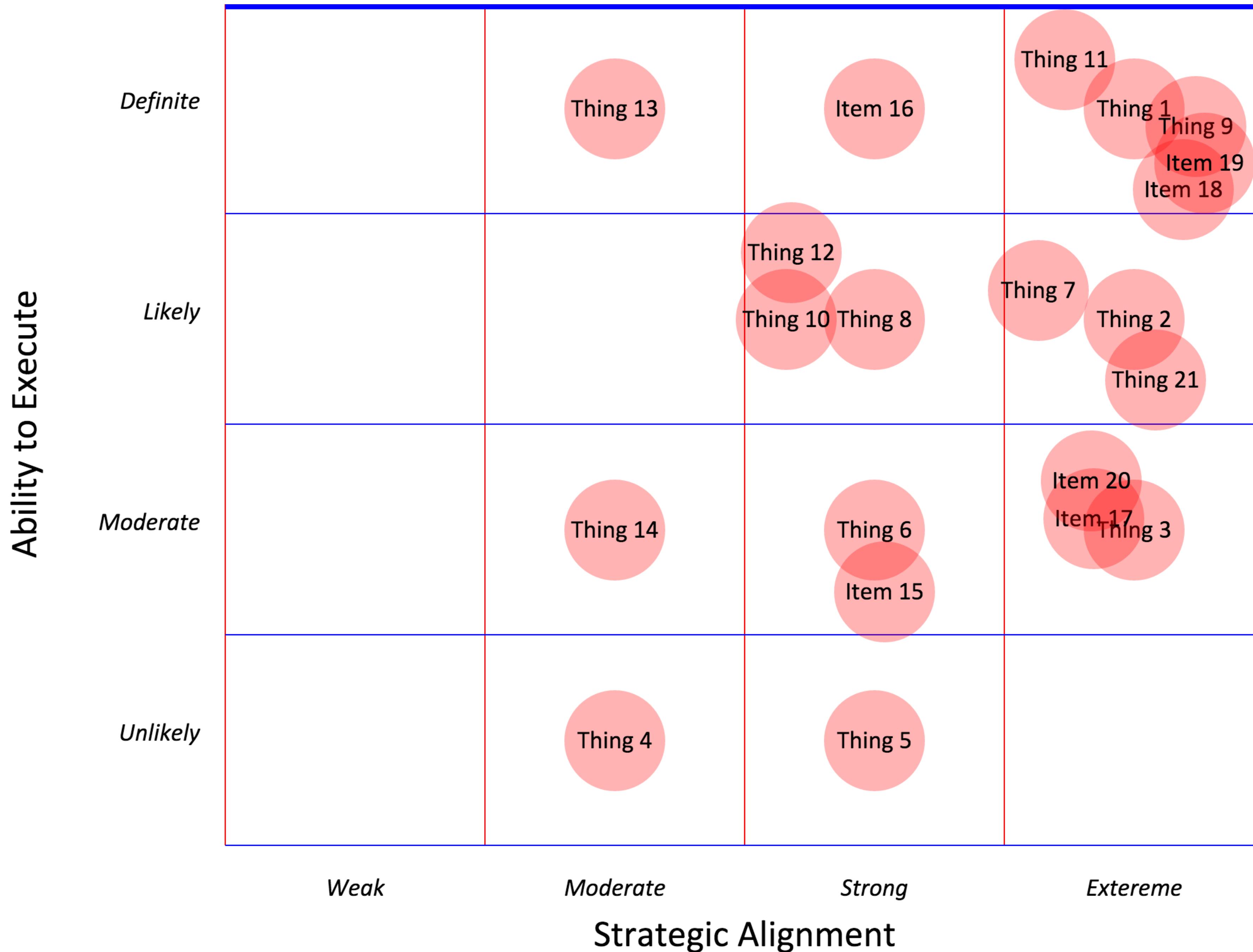
4Q 2010

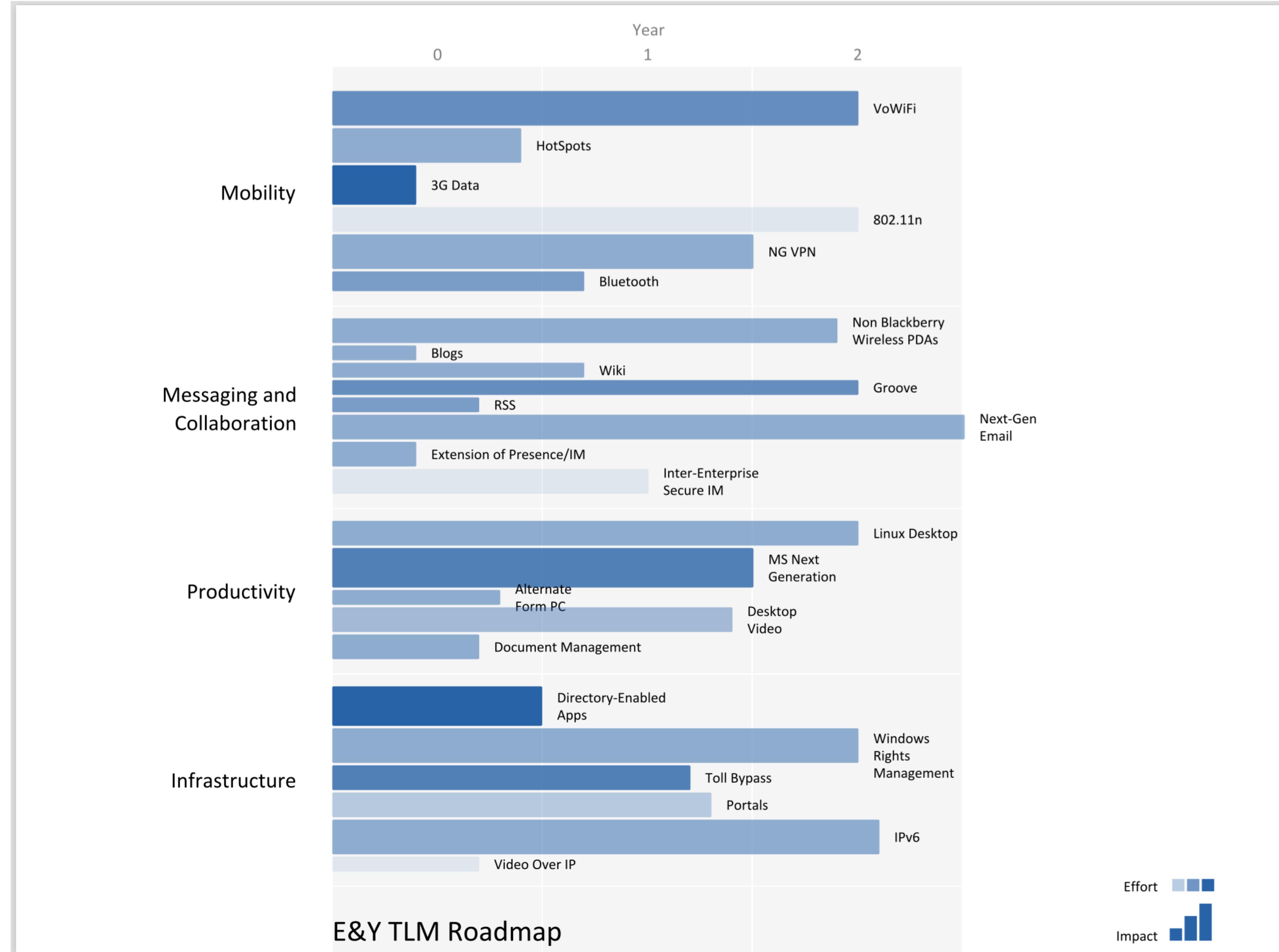


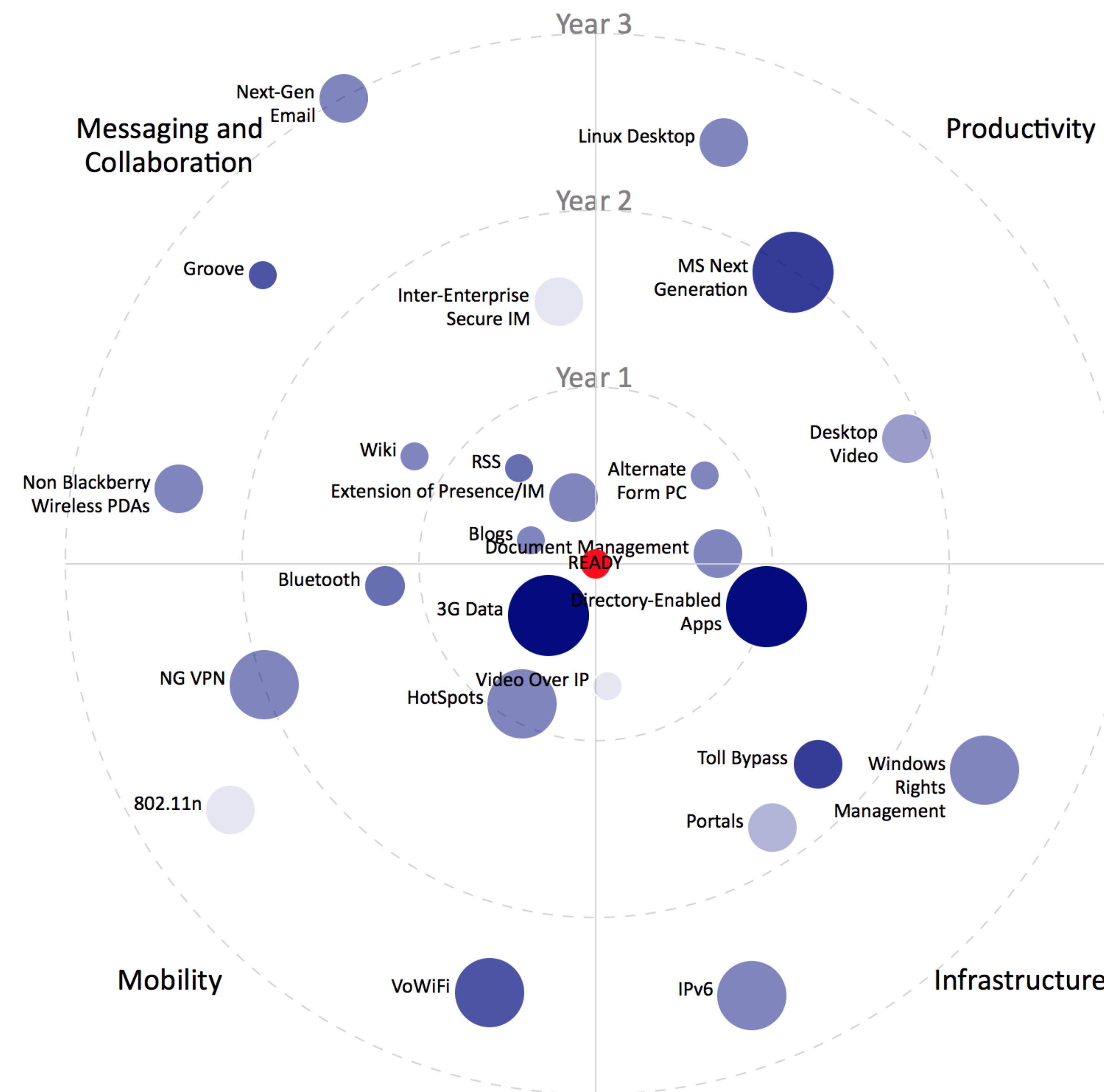


## Sample Bullet Graph

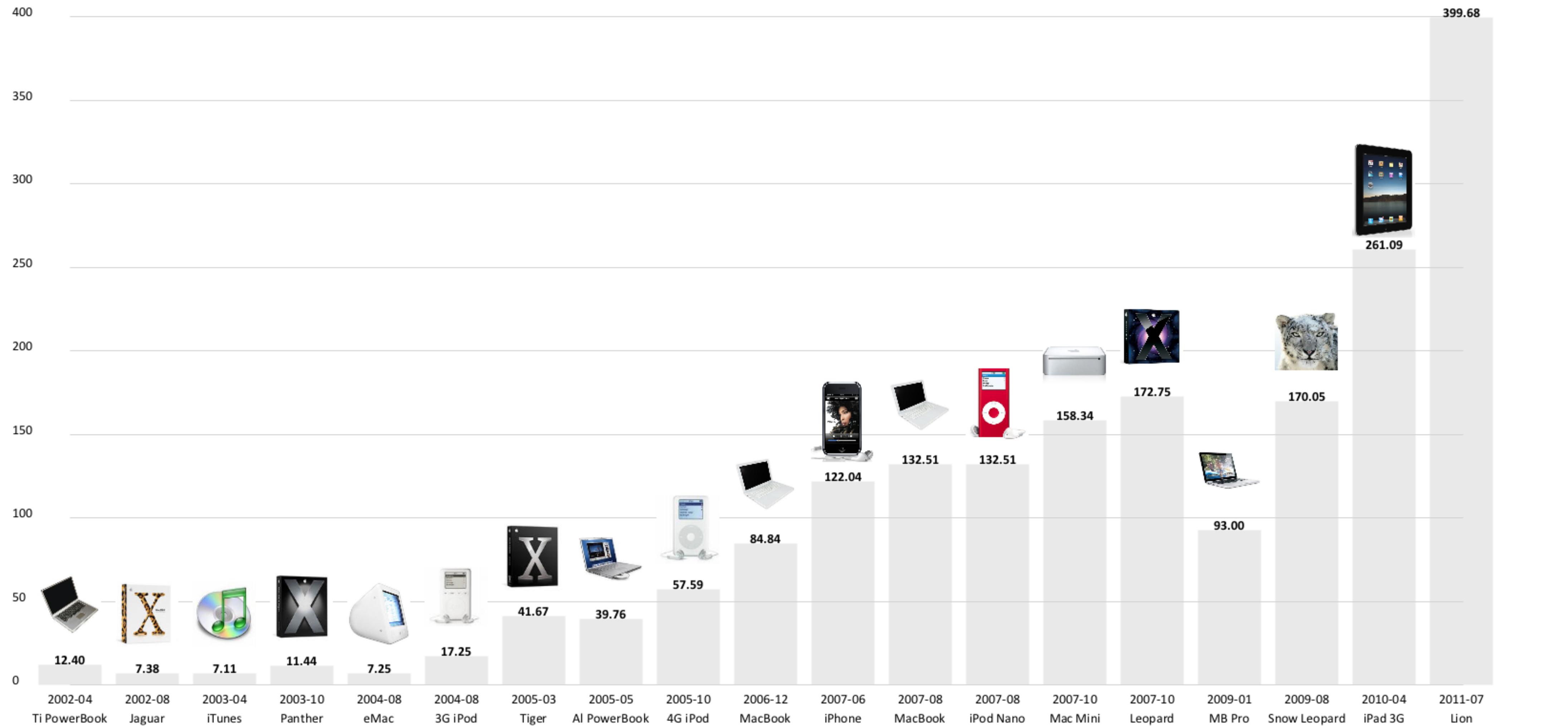
The bullet graph features a single, primary measure (for example, current year-to-date revenue) compares that measure to one or more other measures to enrich its meaning, for example, compared to a target), and displays it in the context of qualitative ranges of performance, such as poor, satisfactory, and good.



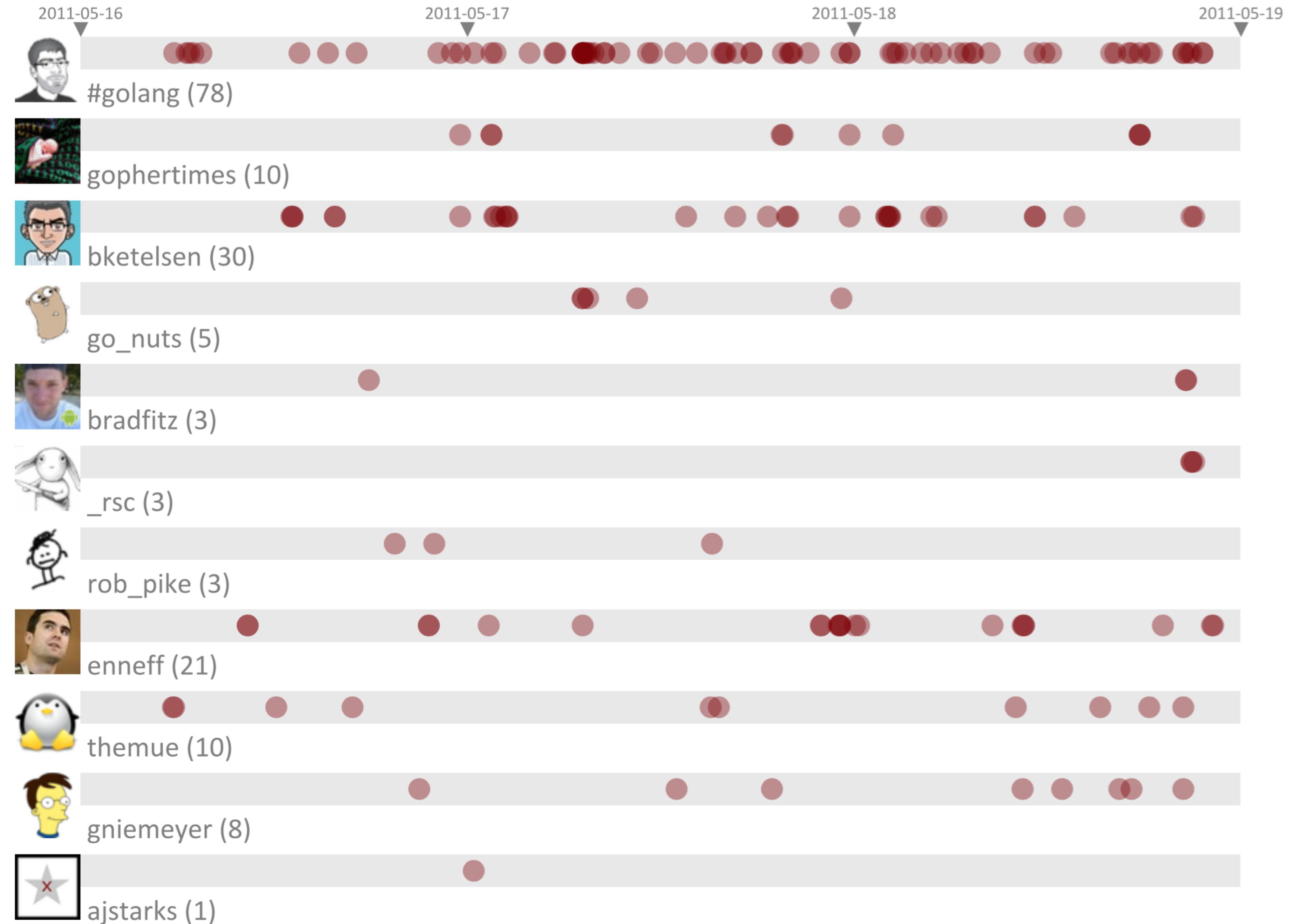




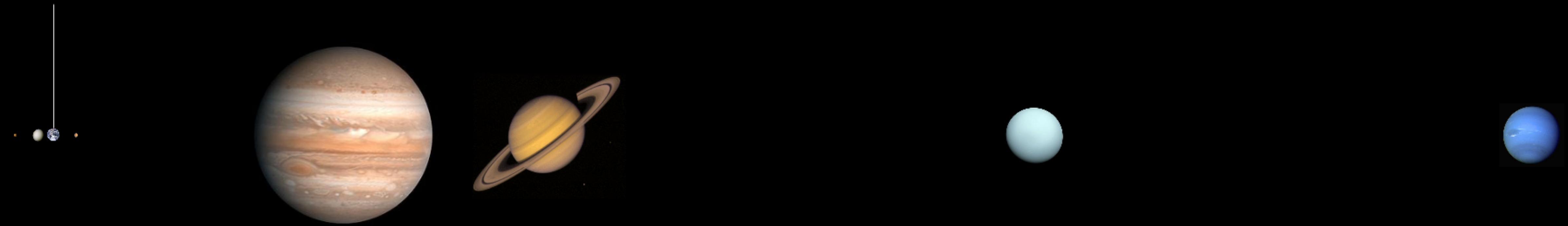
# Apple Purchases and Stock Price



# Twitter Update Frequency



You are here



# JONES

- LAYER TENNIS SEASON 4 WEEK 6 MATCH COMMENTARY BY MIKE MONTEIRO -

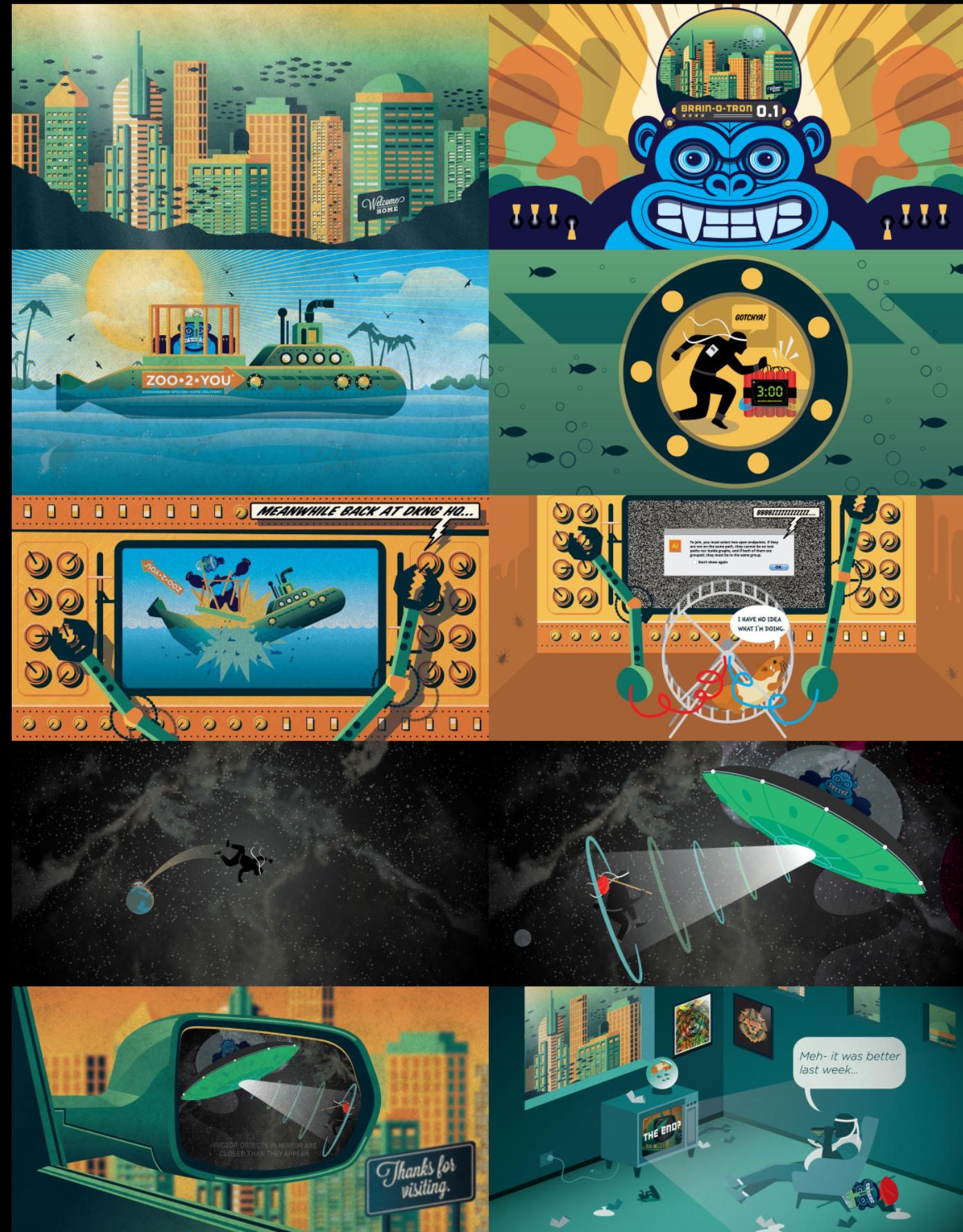
# MONTIEL



# DKNG

- LAYER TENNIS SEASON 4 WEEK 7 MATCH COMMENTARY BY JOHN GRUBER -

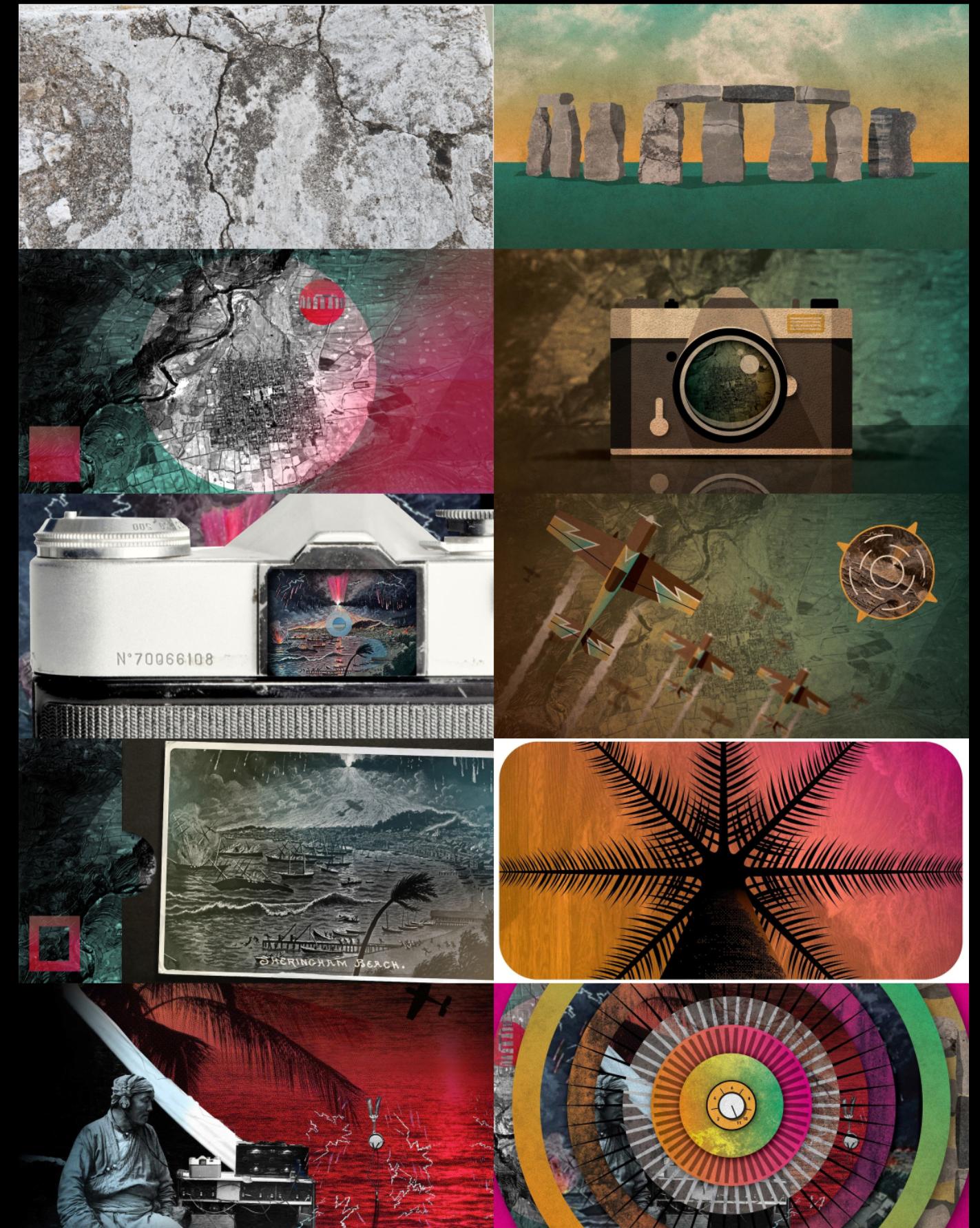
# DDL



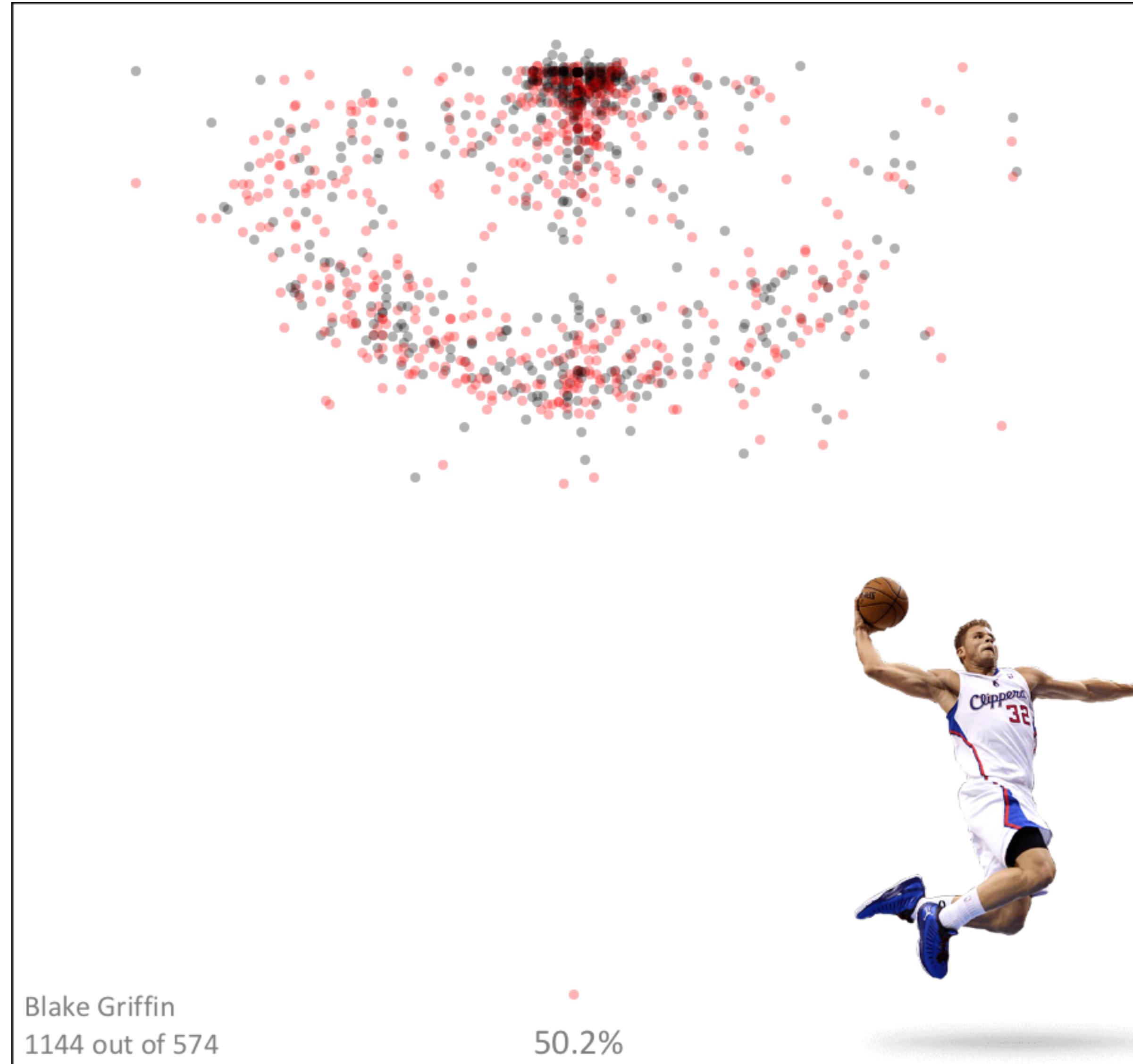
# ANDERSON

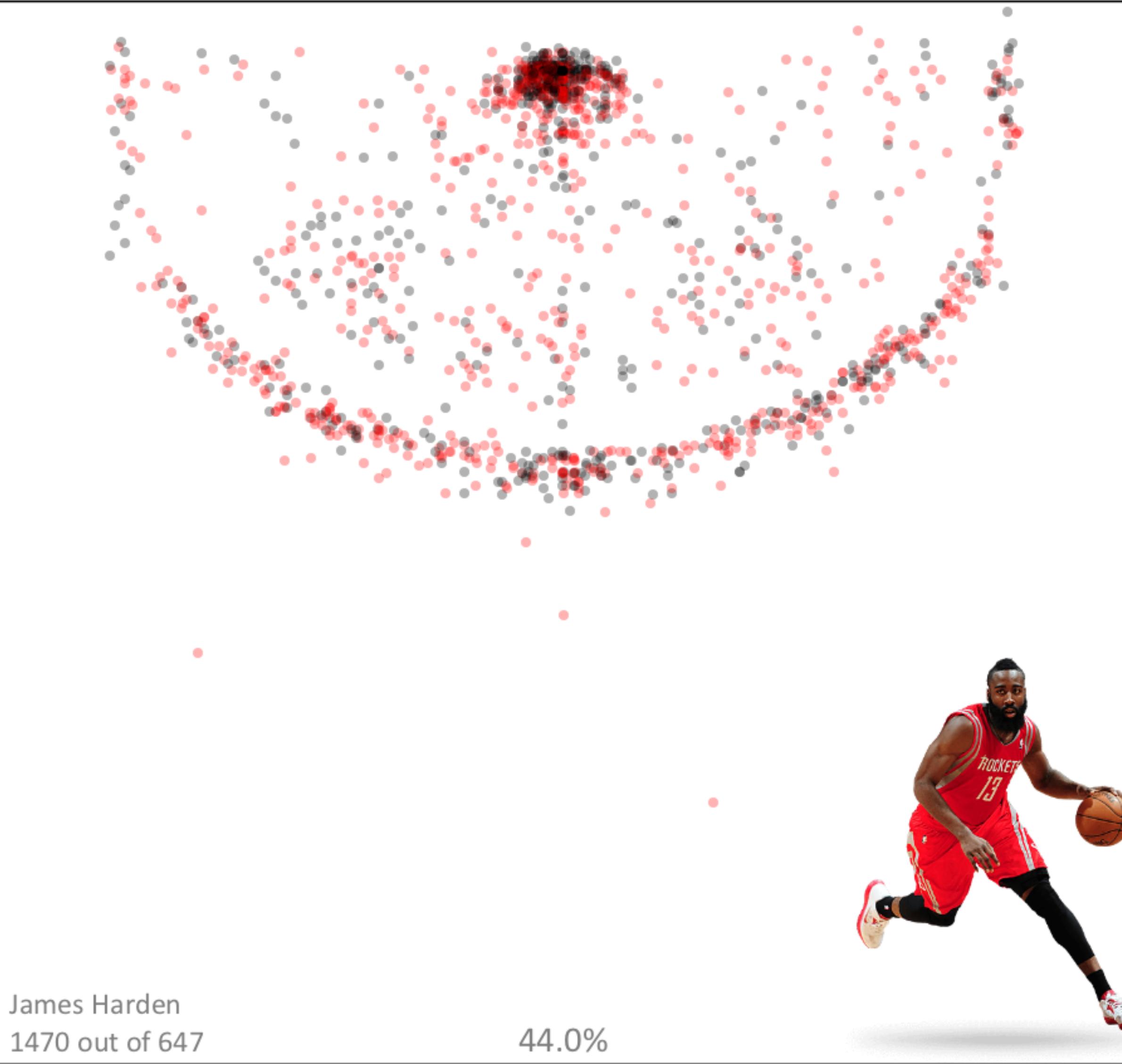
- LAYER TENNIS SEASON 4 PLAYOFF QUARTERFINAL COMMENTARY BY BRYAN BEDELL -

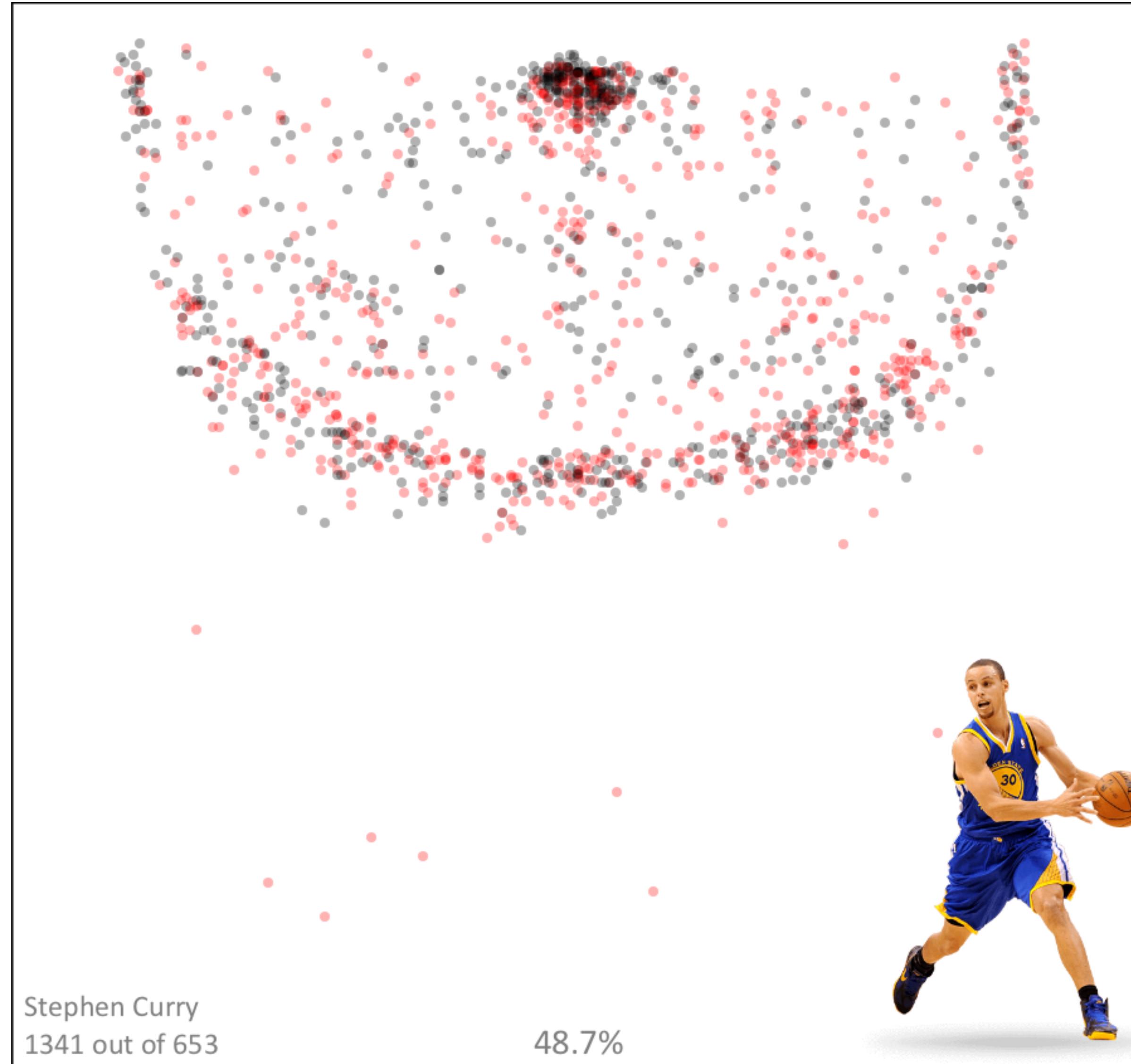
# DKNG

















```
canvas.Def()
canvas.Gid("unit")
canvas.Polyline(xl, yl, "fill:none")
canvas.Polygon(xp, yp)
canvas.Gend()
```

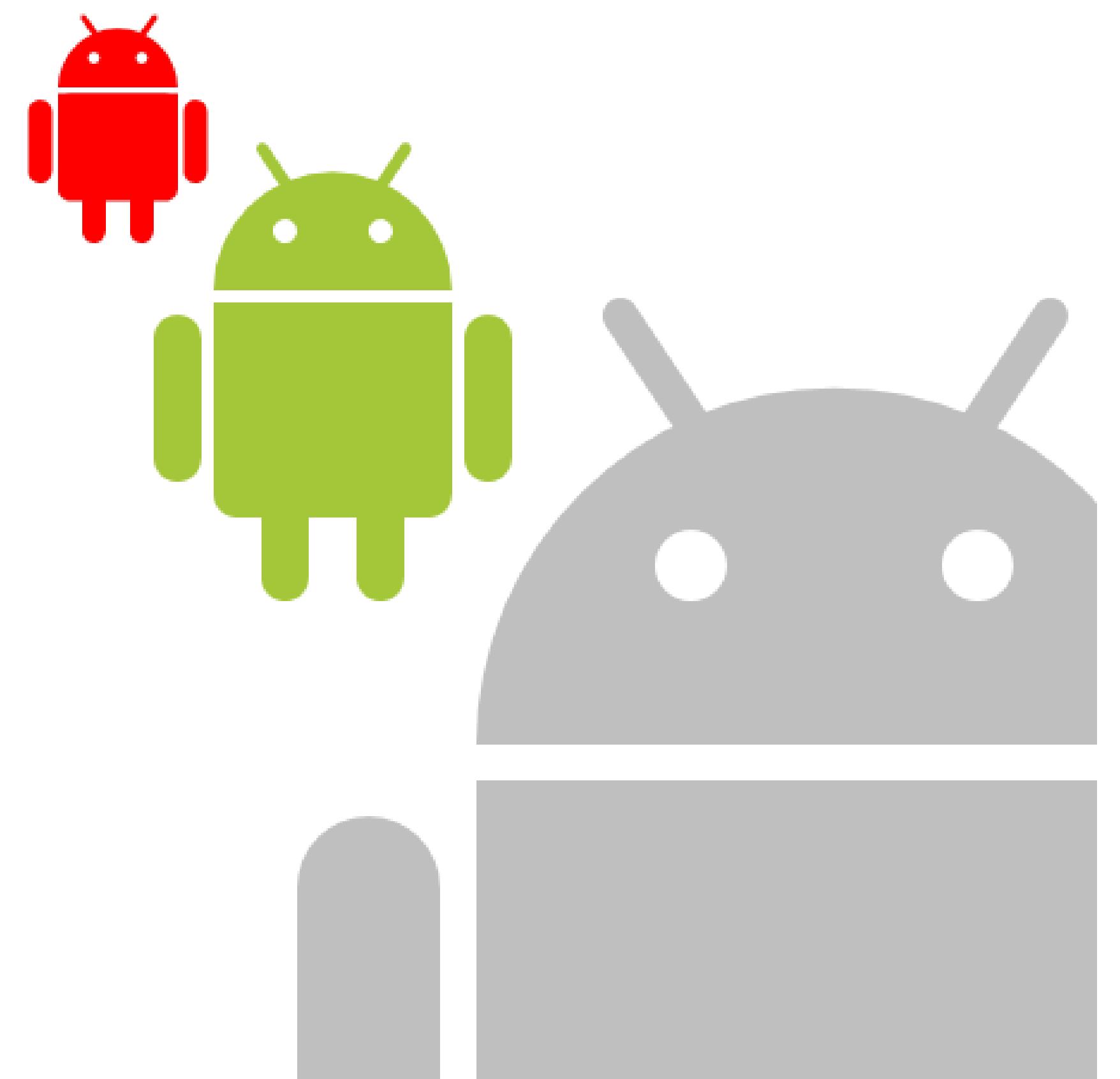


```
canvas.Gid("runit")
canvas.TranslateRotate(150, 180, 180)
canvas.Use(0, 0, "#unit")
canvas.Gend()
canvas.Gend()
canvas.DefEnd()
```



```
for y := 0; < height; y += 130 {
    for x := 100; x < width; x+=100 {
        canvas.Use(x, y, "#unit")
        canvas.Use(x, y, "#runit")
    }
}
```

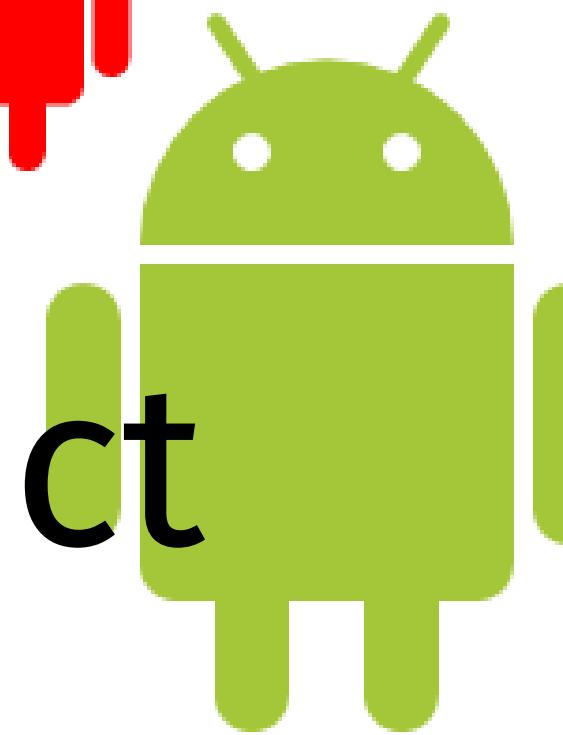
# SVGo API Design



Scale



Roundrect



rgb(164,198,57)

Line

Circle

Arc

Line

Rect

Element

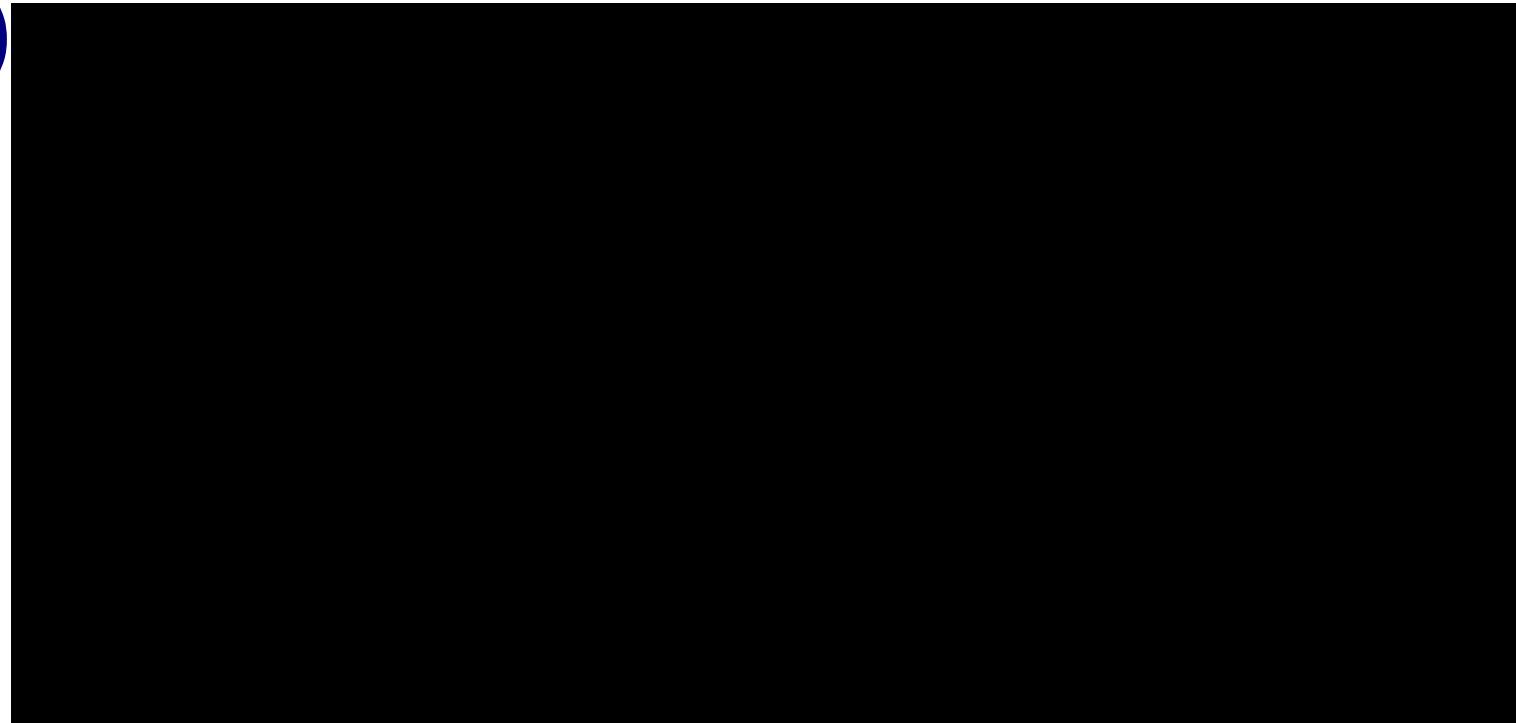
Rect

Arguments

(100,200,250,125)

```
<rect x="100" y="200" width="250" height="125"/>
```

(100, 200)



125

250

Element	Arguments	CSS Style
Rect	(100,200,250,125, "fill:gray;stroke:blue")	

```
<rect x="100" y="200" width="250" height="125"  
      style="fill:gray;stroke:blue"/>
```

(100, 200)



125

250

Element

Rect

Arguments

(100,200,250,125,

`id='box'`, `fill='gray'`, `stroke='blue'`)

Attributes

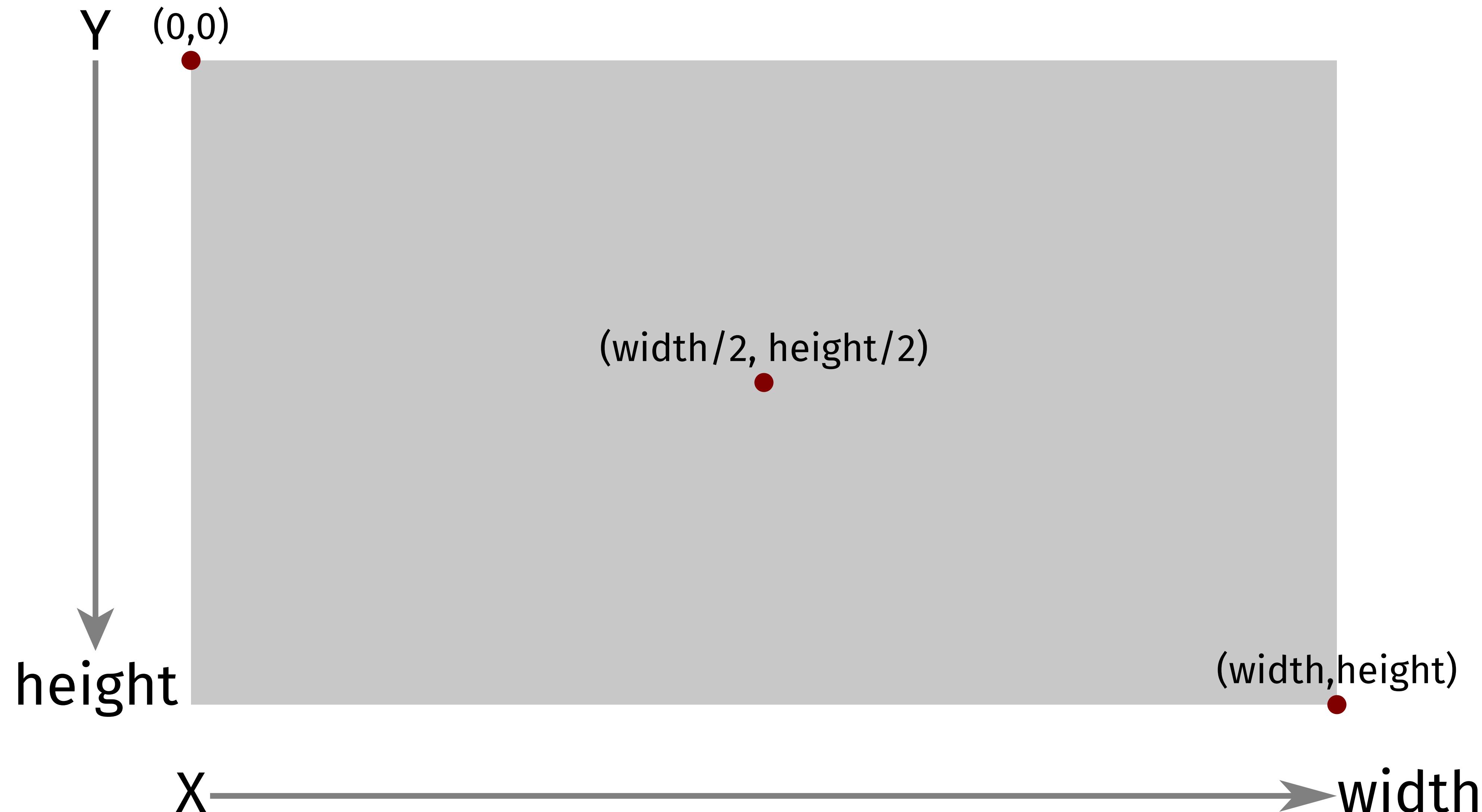
```
<rect x="100" y="200" width="250" height="125"  
      id='box' fill='gray' stroke='blue'/>
```

(100, 200)



125

250



Rect(0,0,width,height)

hello,  
world

(width/2, height/2)

(width/2, height)

```
package main

import (
    "os"
    "github.com/ajstarks/svg"
)

func main() {
    width := 960
    height := 540
    style := "fill:white;font-size:60pt;text-anchor:middle"
    canvas := svg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Circle(width/2, height, width/2, "fill:rgb(44,77,232)")
    canvas.Text(width/2, height/2, "hello, world", style)
    canvas.End()
}
```



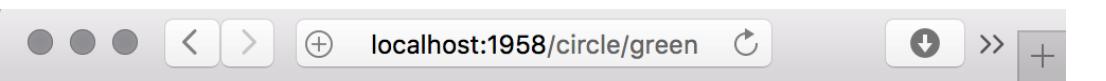
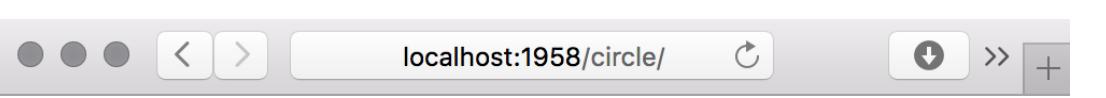
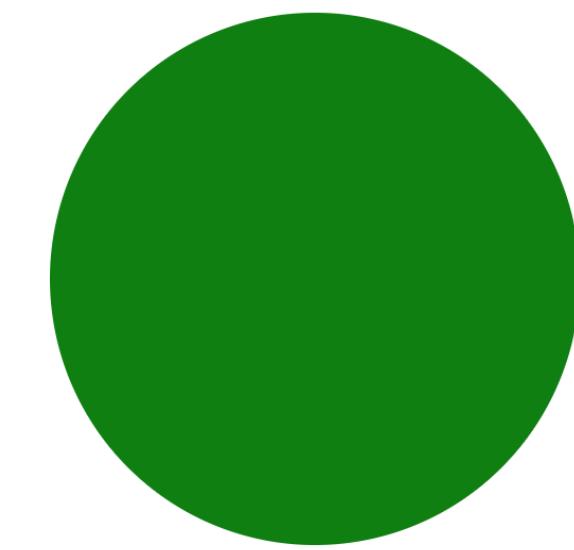
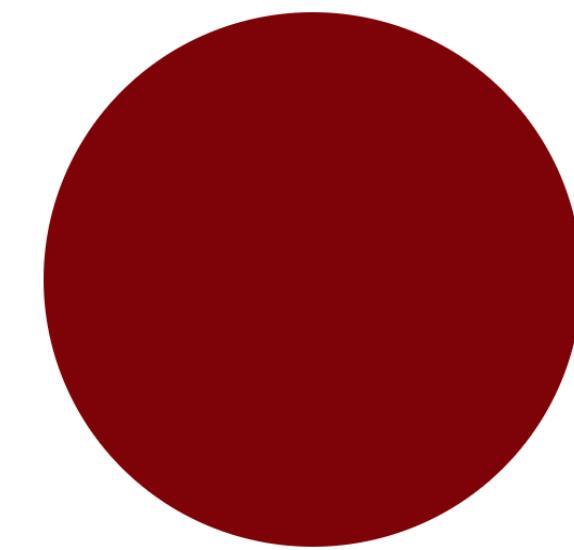
/circle

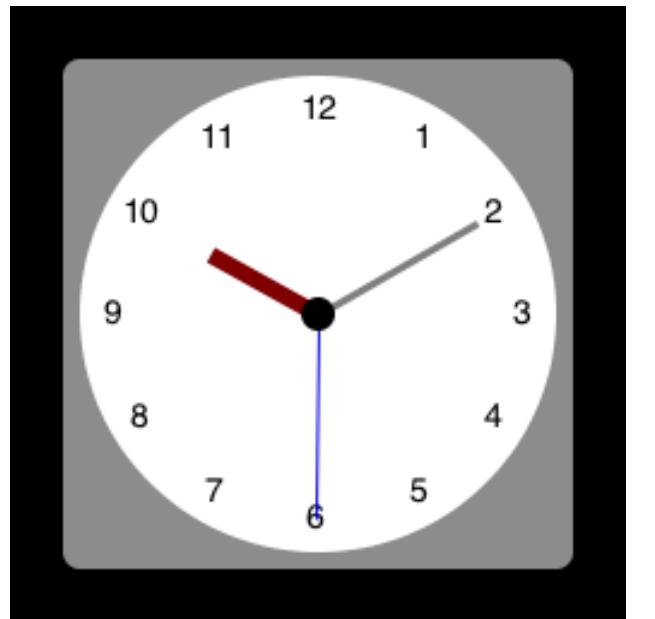
```
const defaultstyle = "fill:rgb(127,0,0)"

func circle(w http.ResponseWriter, req *http.Request) {
    w.Header().Set("Content-Type", "image/svg+xml")
    s := svg.New(w)
    s.Start(500, 500)
    s.Title("Circle")
    s.Circle(250, 250, 125, shapestyle(req.URL.Path))
    s.End()
}
```

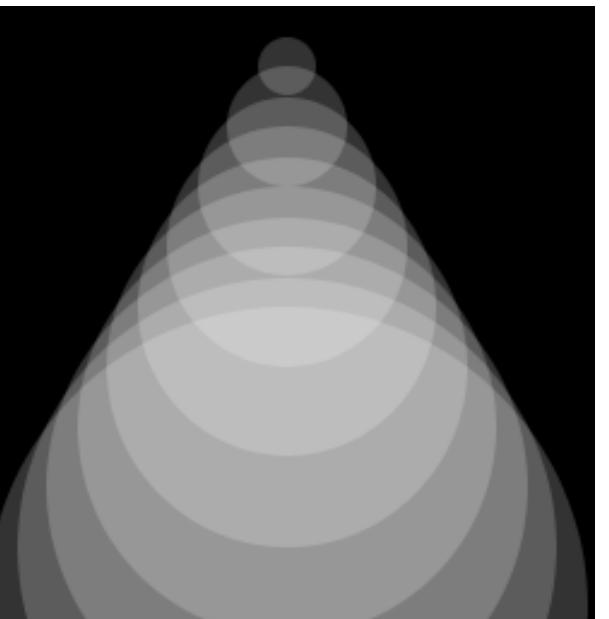
/circle/green

```
func shapestyle(path string) string {
    i := strings.LastIndex(path, "/") + 1
    if i > 0 && len(path[i:]) > 0 {
        return "fill:" + path[i:]
    }
    return defaultstyle
}
```

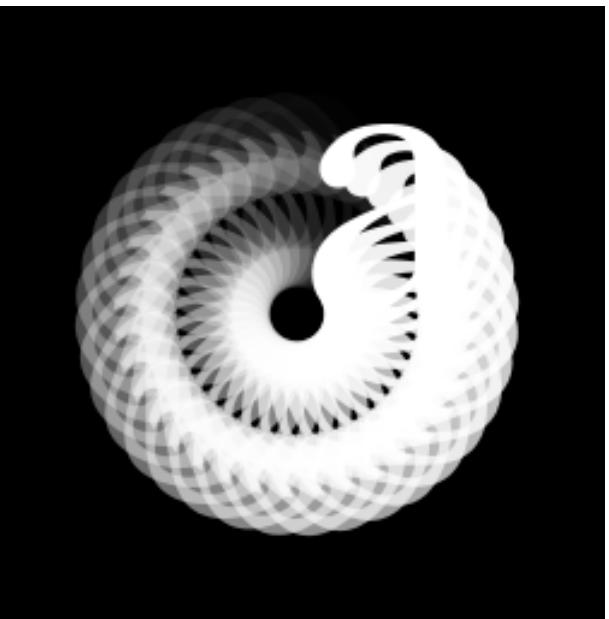




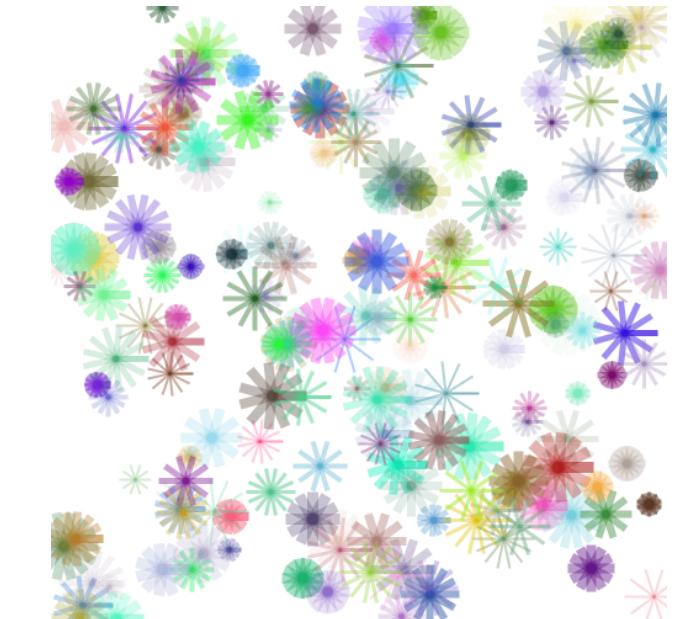
clock



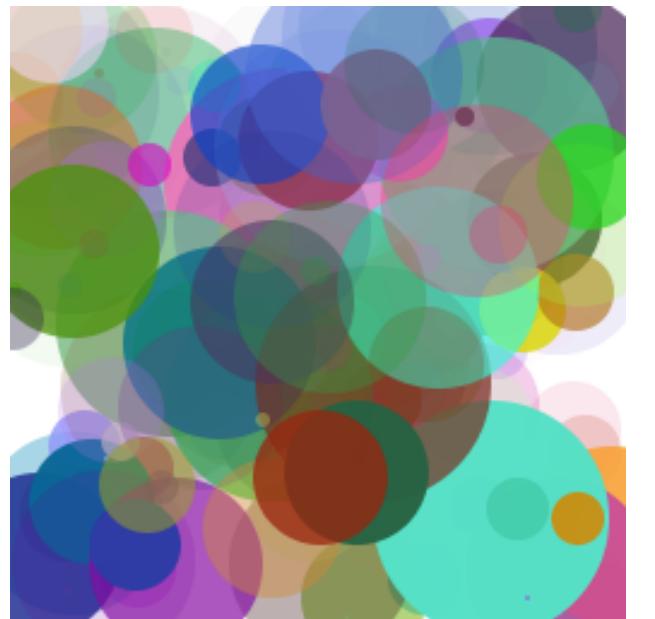
funnel



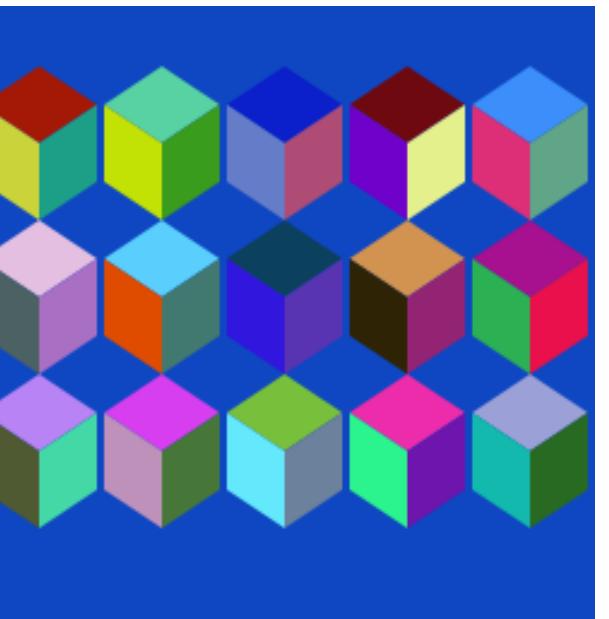
rotext



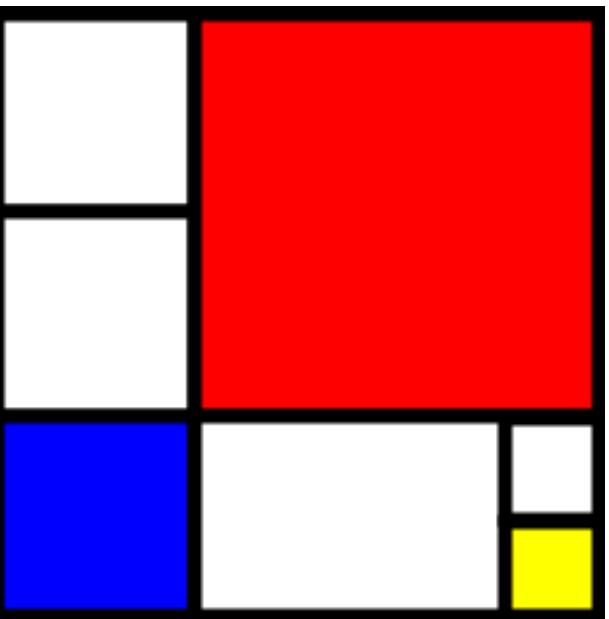
flower



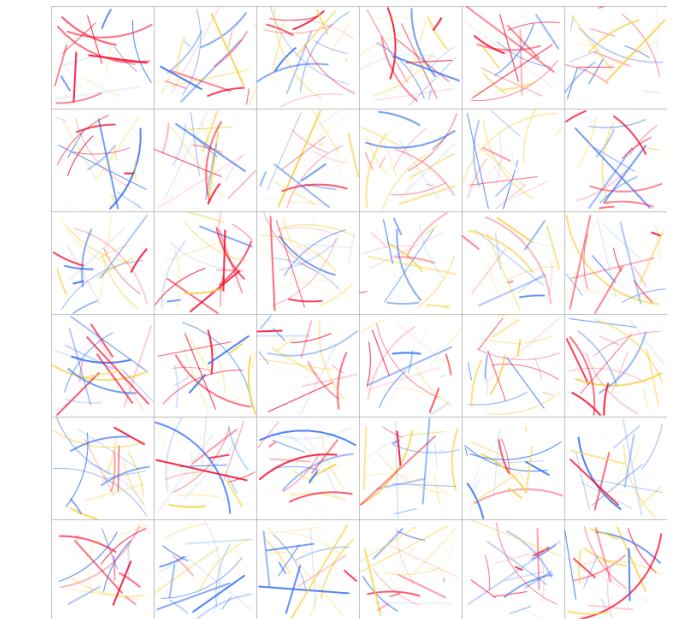
rshape



cube



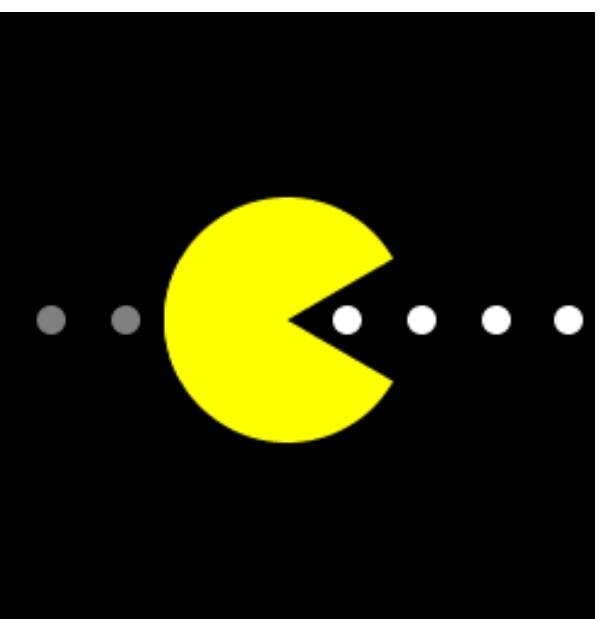
mondrian



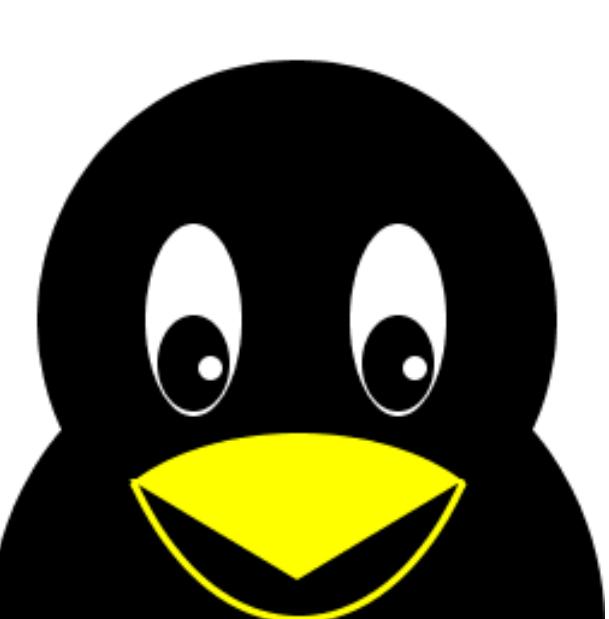
lewitt



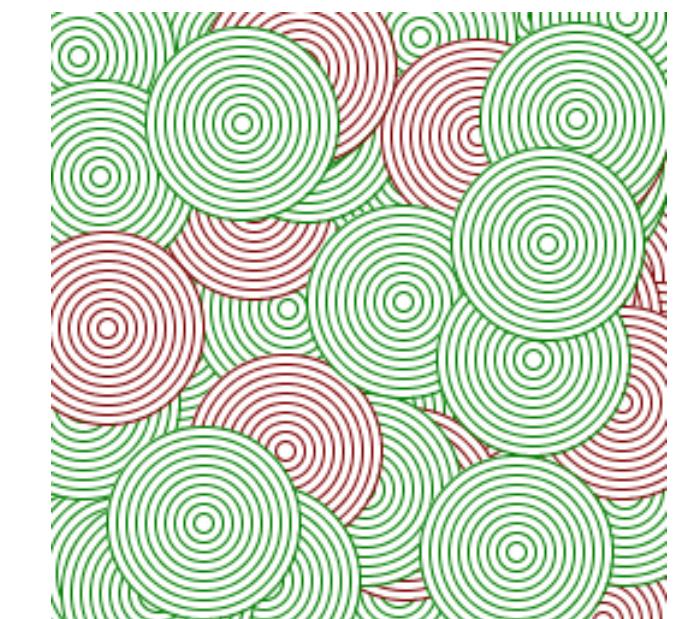
face



pacman

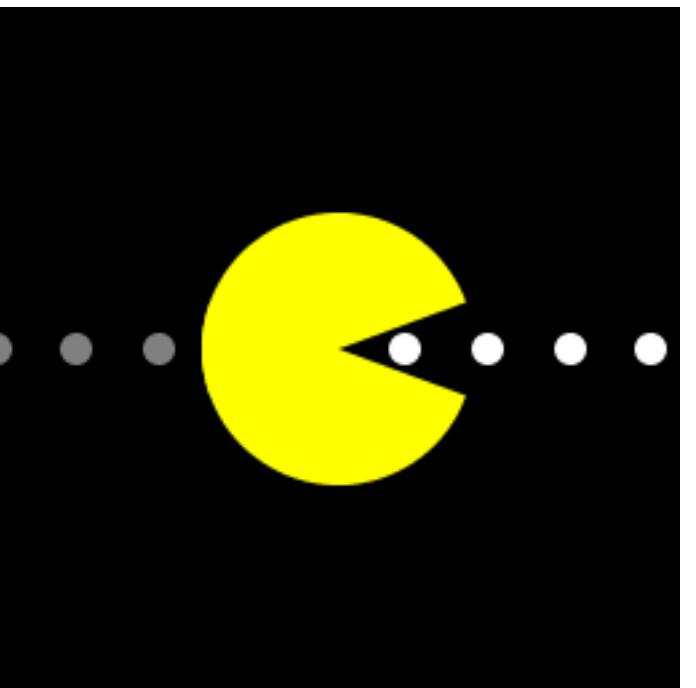


tux

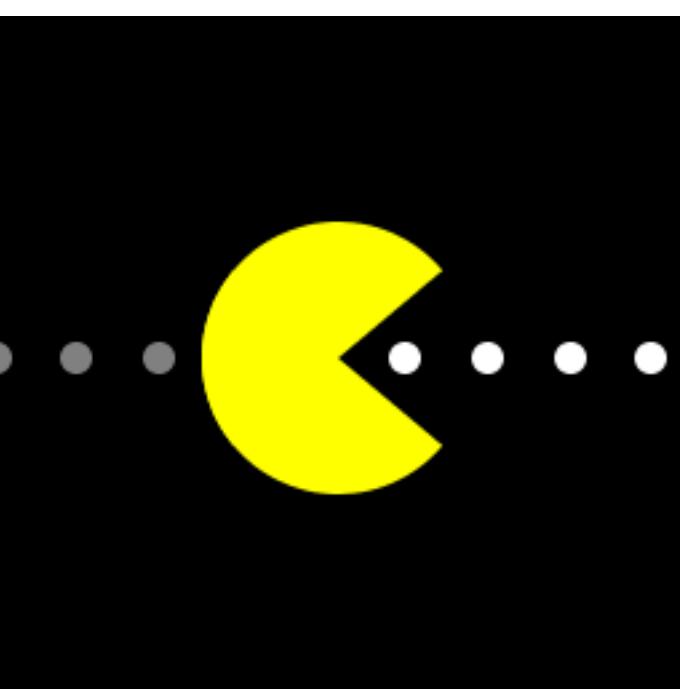


concentric

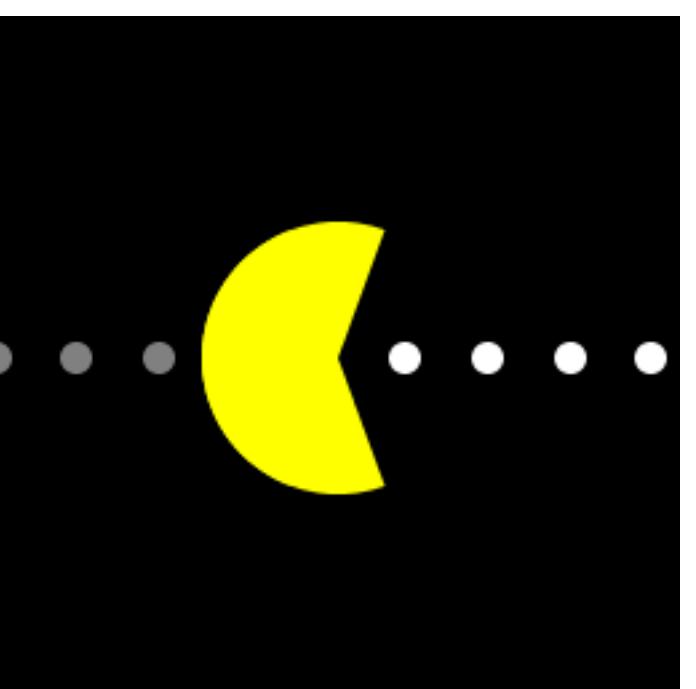
/pacman/?angle=20



/pacman/?angle=40



/pacman/?angle=70

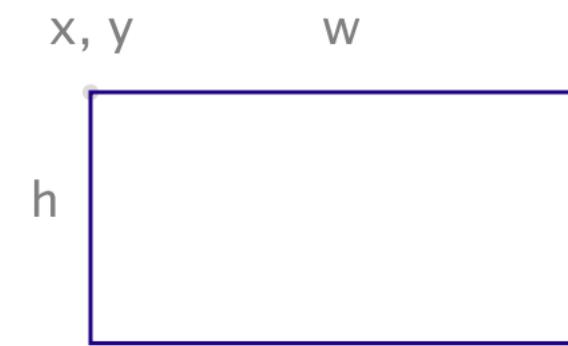


# API Reference

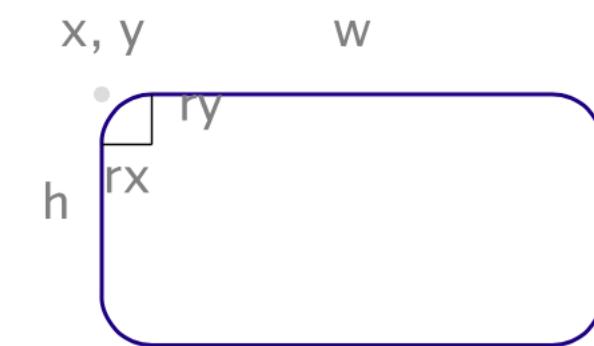
# SVGo Method Categories

- Shapes
- Paths, Lines and Curves
- Text and Images
- Colors and Gradients
- Transforms
- Skewing
- Filter Effects
- Starting and Ending
- Metadata
- Utility
- Groups and Definitions
- Markers and Patterns
- Links
- Masking and Clipping

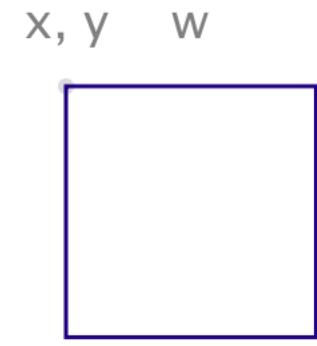
# Shapes



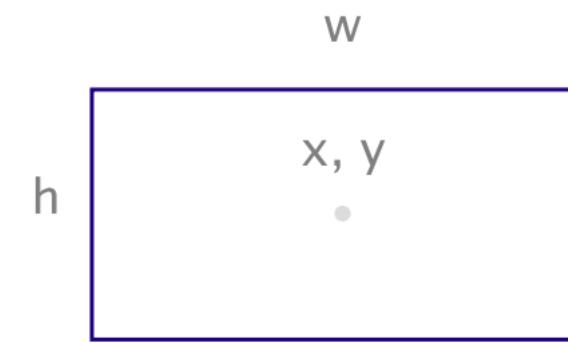
Rect(x, y, w, h int, s string...)



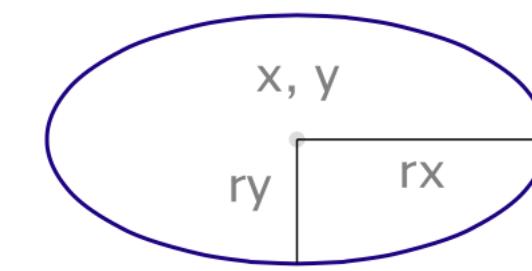
Roundrect(x, y, w, h, rx, ry int, s string...)



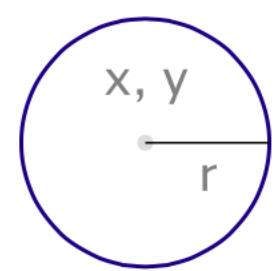
Square(x, y, l int, s string...)



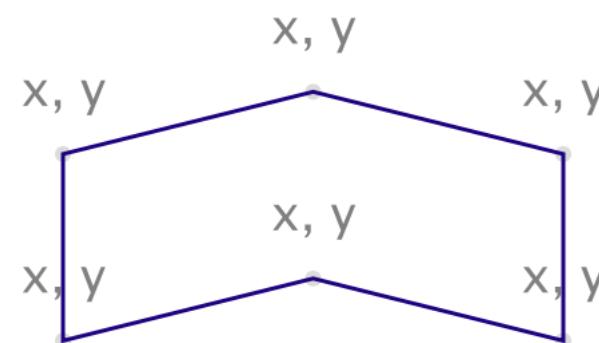
CenterRect(x, y, w, h int, s string...)



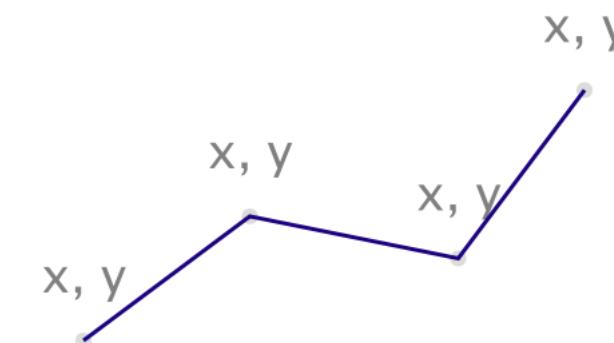
Ellipse(x, y, w, h int, s string...)



Circle(x, y, r int, s string...)



Polygon(x, y []int, s string...)



Polyline(x, y []int, s string...)

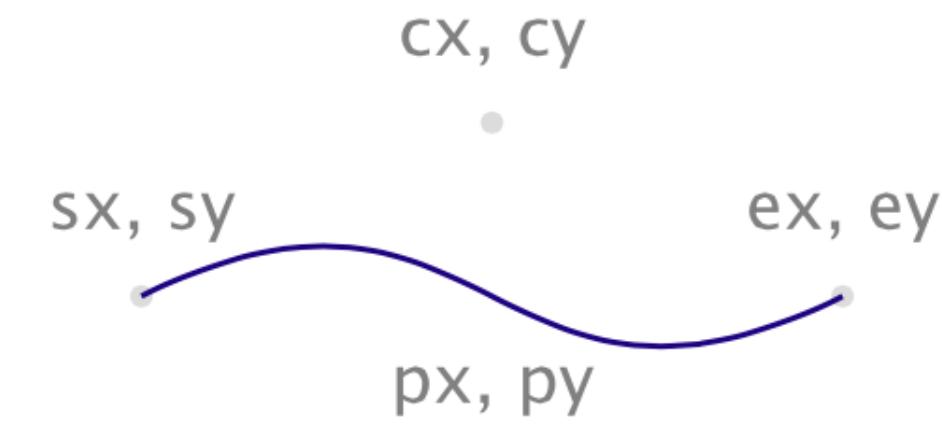


Line(x1, y2, x2, y2 int, s string...)

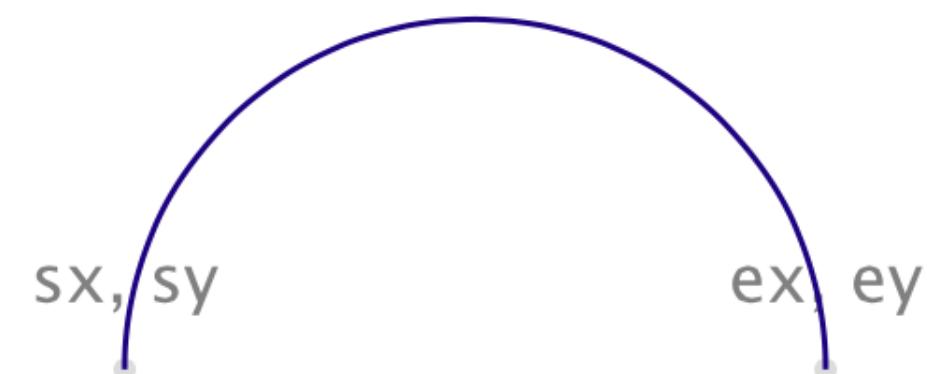
# Curves and Paths



`Bezier(sx, sy, cx, cy, ex, ey int, s string...)`



`QBez(sx, sy, cx, cy, px, py, ex, ey int, s string...)`



`Arc(sx, ax, ay, r int, dir, large bool, ex, ey int, s string...)`



`Path(d string, s string...)`

# Text and Images

hello, this is SVG  
x, y

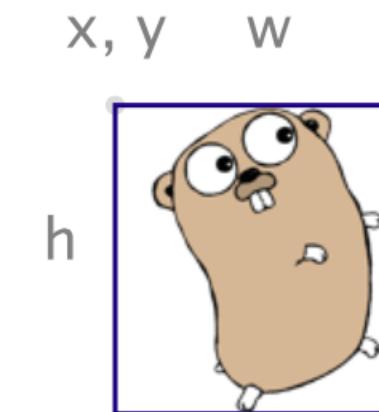
Text(x, y int, s string, s ...string)

It's "fine" & "dandy" to draw text along

Textpath(t string, pathid string, s ...string)

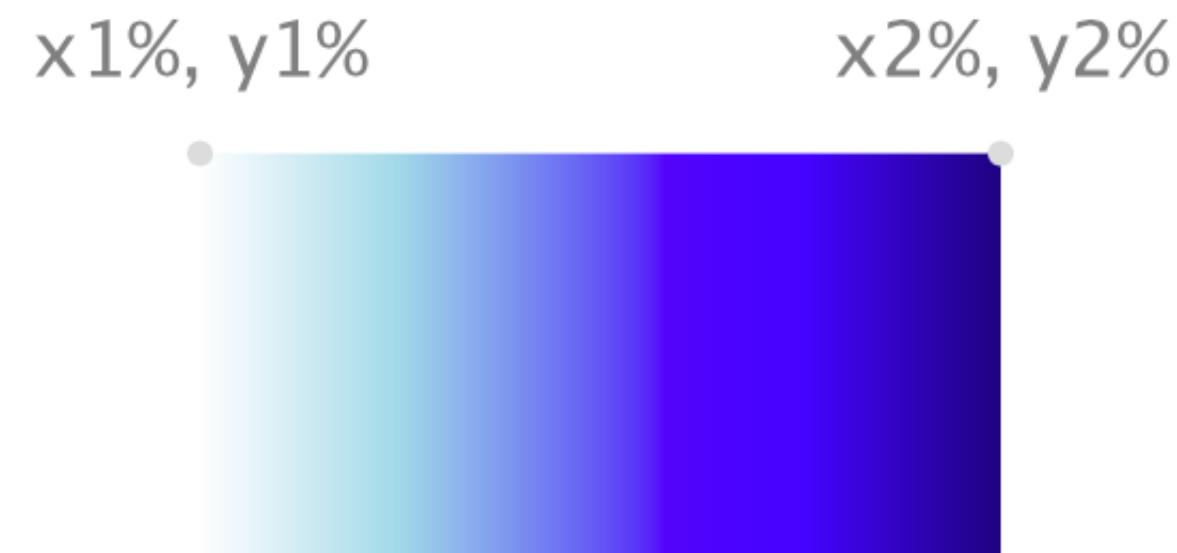
```
New(w io.Writer)  
Start(w, h int, options ...string)/End()  
Startview(w, h, minx, miny, vw, vh int)  
Group(s ...string)/Gend()  
Gstyle(s string)/Gend()
```

Textlines(x, y int, s []string, size, spacing int, fill, align string)

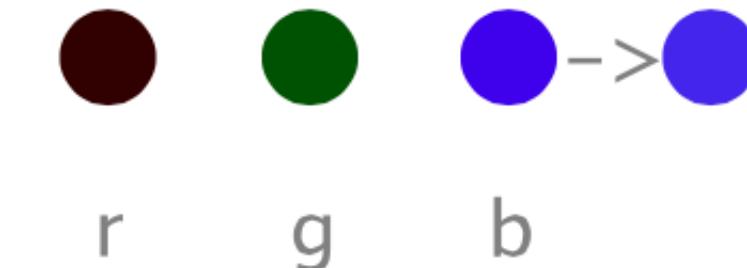


Image(x, y, w, h int, link string, s ...string)

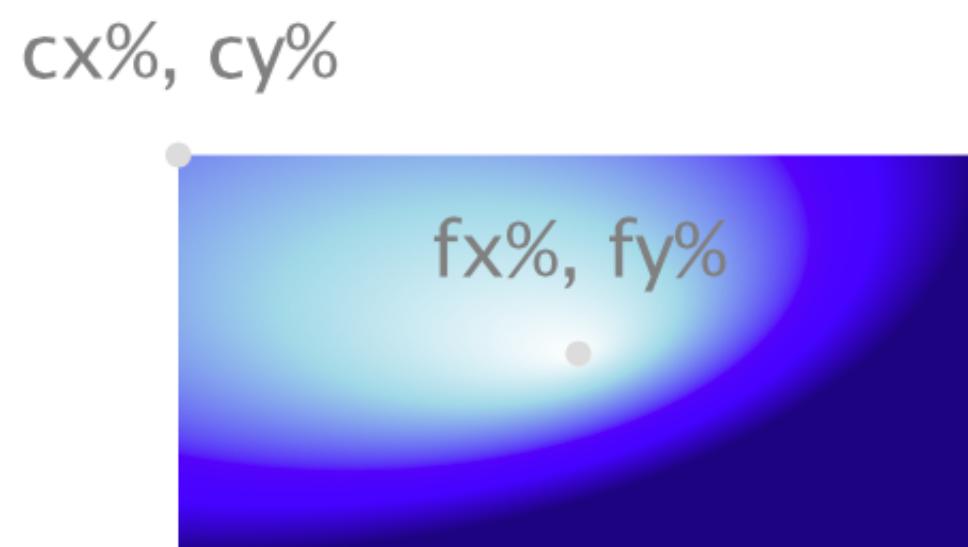
# Gradients and Colors



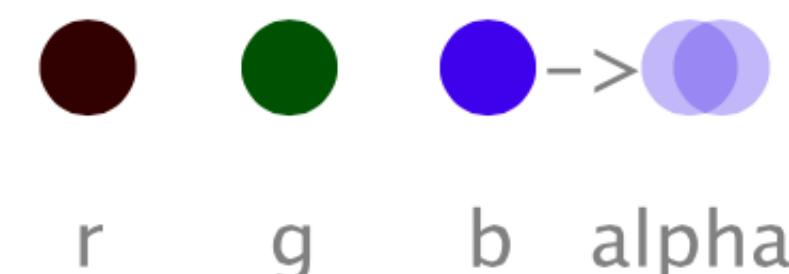
LinearGradient(id string, x1, y1, x2, y2 uint8, sc []Offcolor)



RGB(r, g, b int)

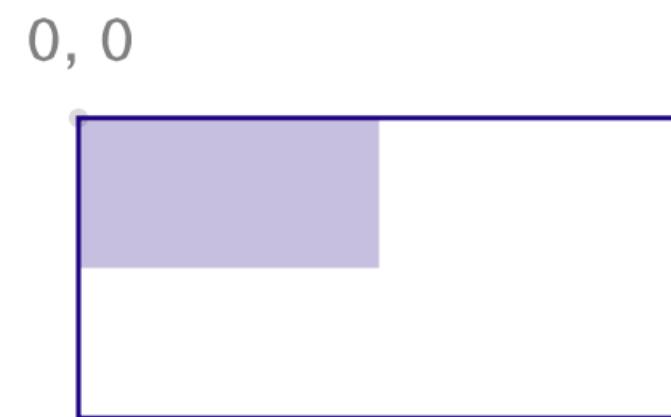


RadialGradient(id string, cx, cy, r, fx, fy uint8, sc []Offcolor)

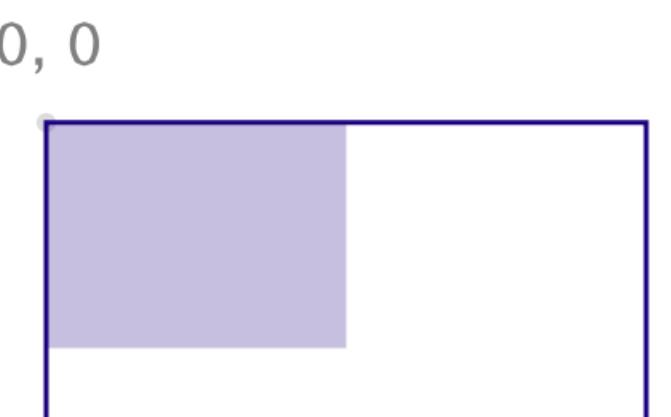


RGBA(r, g, b int, a float64)

# Transforms



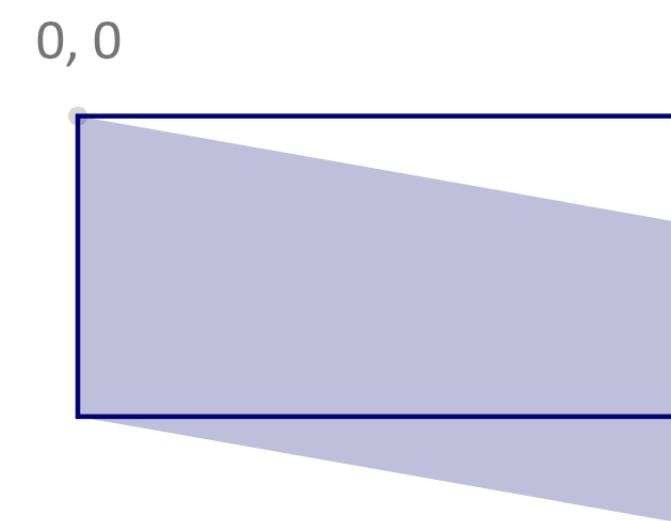
Scale(n float64)



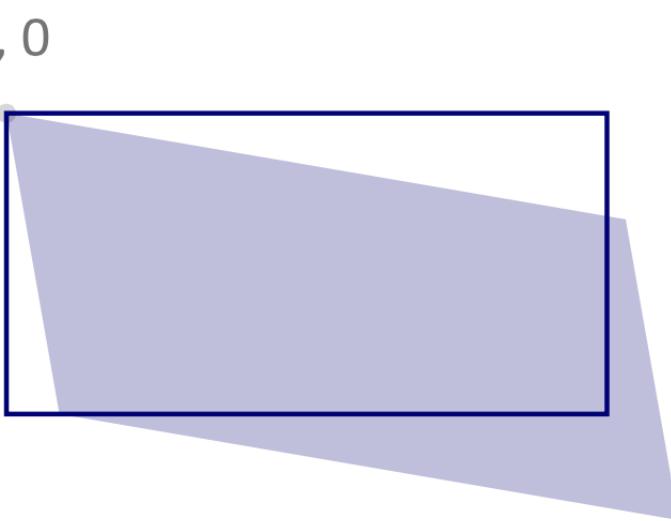
ScaleXY(dx, dy float64)



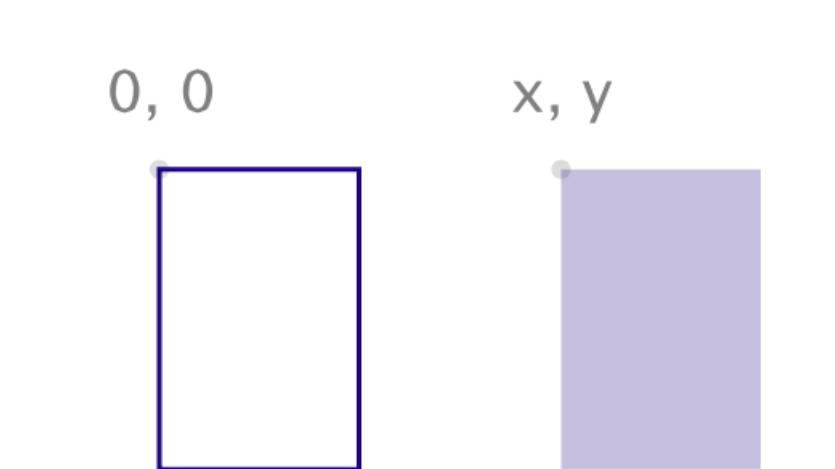
SkewX(a float64)



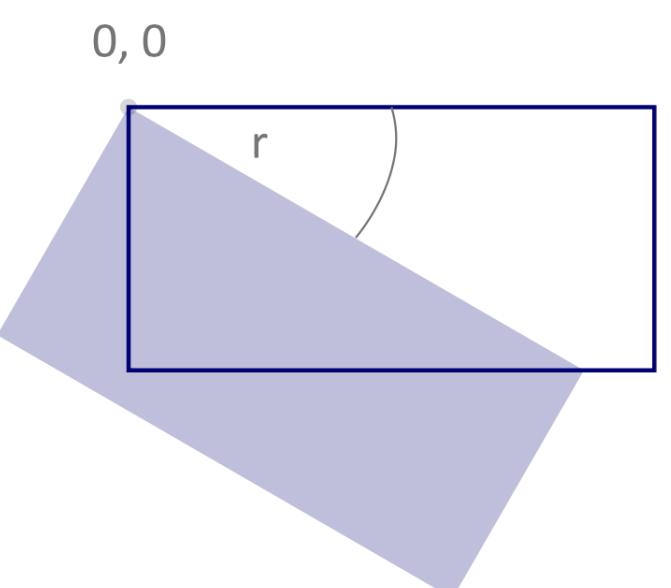
SkewY(a float64)



SkewXY(ax, ay float64)

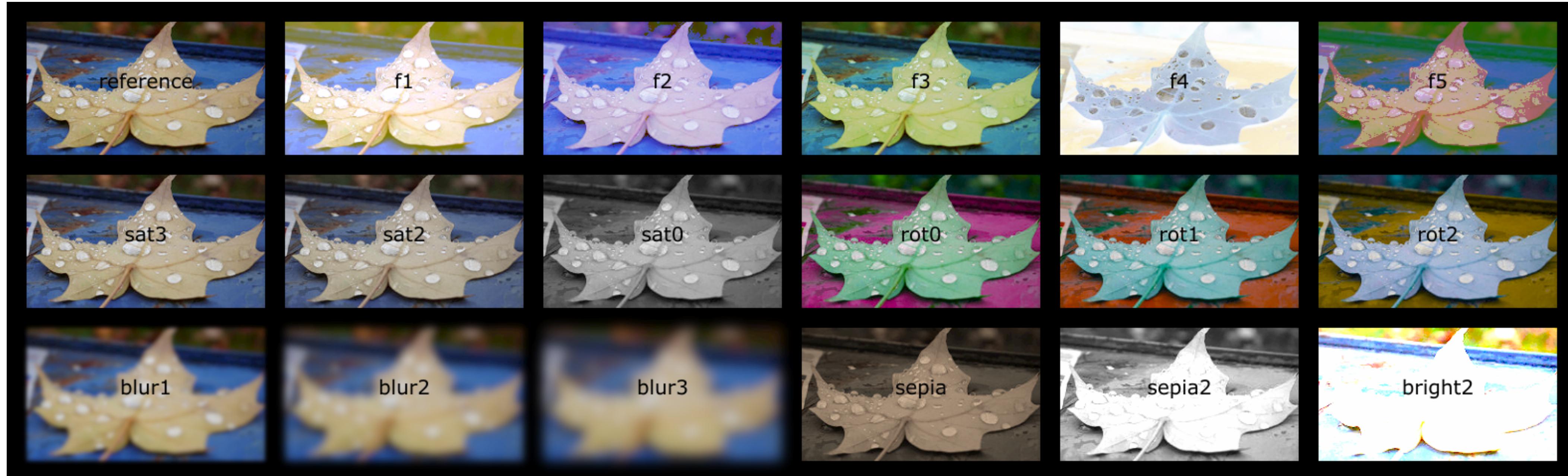


Translate(x, y int)



Rotate(a float64)

# Filter Effects



# Sketching with code

```
package main

import (
    "os"
    "github.com/ajstarks/svg"
)

func main() {
    width := 960
    height := 540
    style := "fill:white;font-size:60pt;text-anchor:middle"
    canvas := svg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Circle(width/2, height, width/2, "fill:rgb(44,77,232)")
    canvas.Text(width/2, height/2, "hello, world", style)
    canvas.End()
}
```



(Shift-Enter to compile and run.)

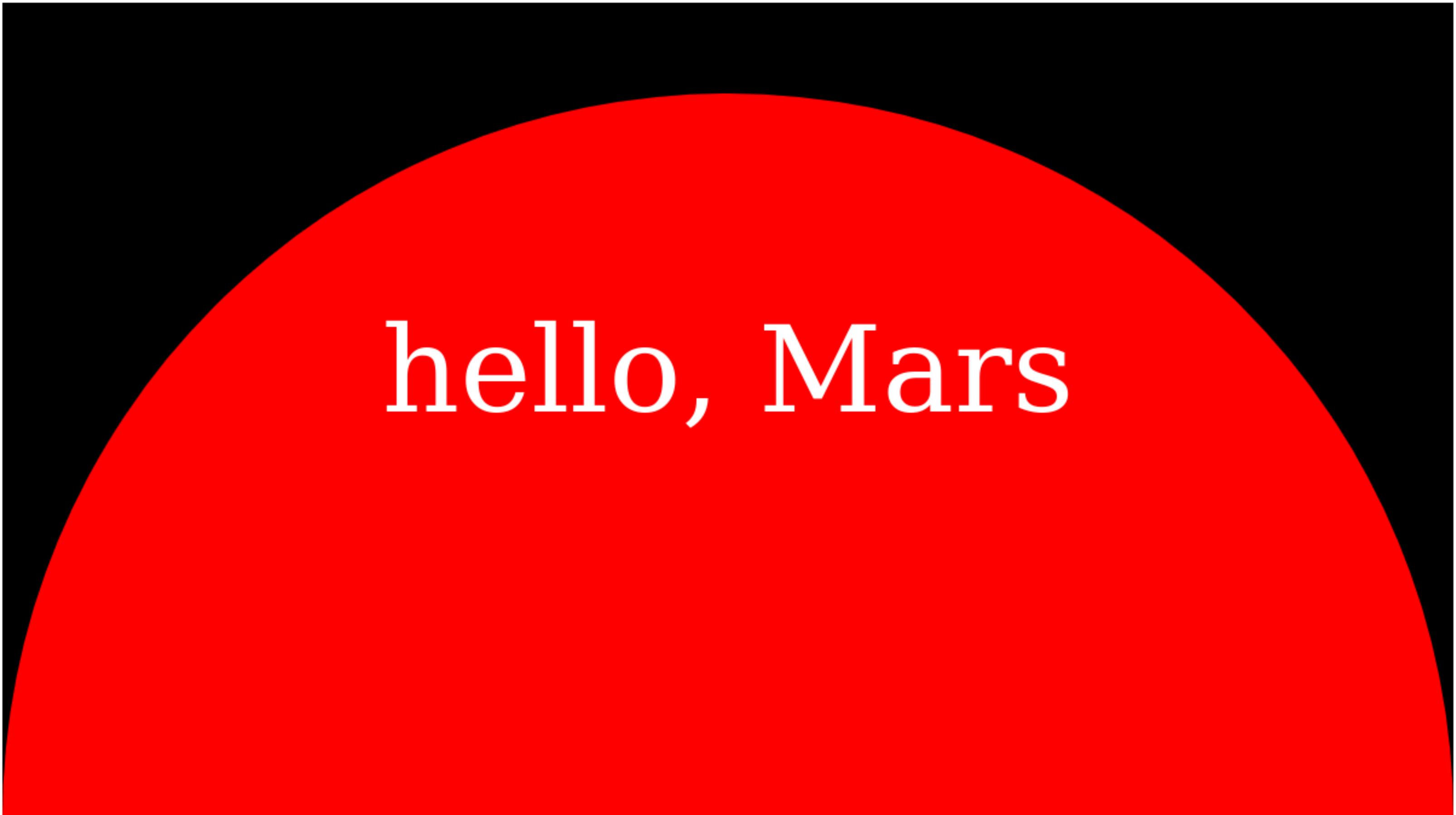


Compile and run after each keystroke

```
package main

import (
    "os"
    "github.com/ajstarks/svg"
)

func main() {
    width := 960
    height := 540
    style := "fill:white;font-size:60pt;text-anchor:middle"
    canvas := svg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Circle(width/2, height, width/2, "fill:red")
    canvas.Text(width/2, height/2, "hello, Mars", style)
    canvas.End()
}
```



(Shift-Enter to compile and run.)



Compile and run after each keystroke

svgplay: SVGo sketching × +

localhost:1999

Apps PROTOTYPE -events | Data... Other bookmarks

```
package main

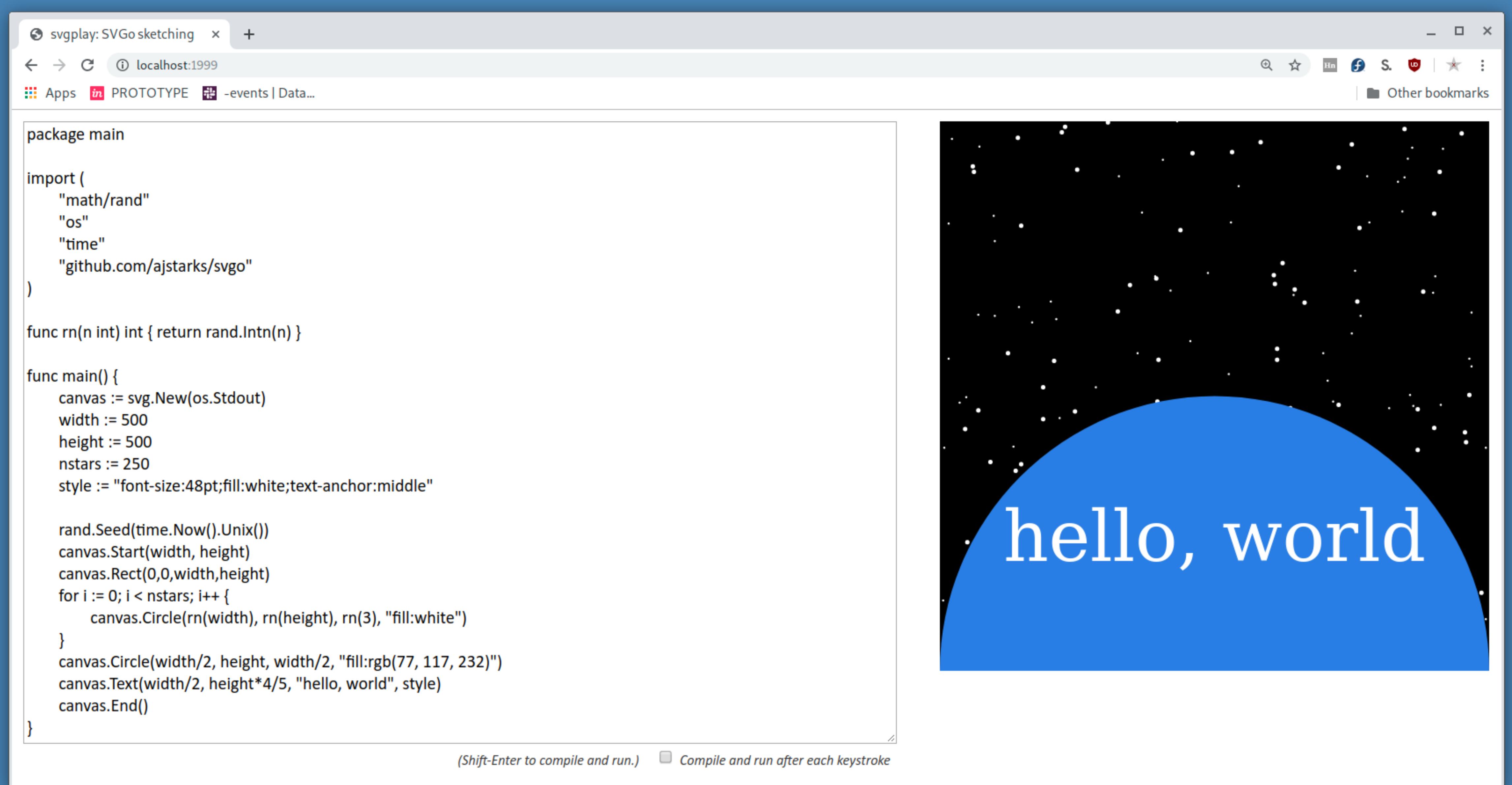
import (
    "math/rand"
    "os"
    "time"
    "github.com/ajstarks/svg"
)

func rn(n int) int { return rand.Intn(n) }

func main() {
    canvas := svg.New(os.Stdout)
    width := 500
    height := 500
    nstars := 250
    style := "font-size:48pt;fill:white;text-anchor:middle"

    rand.Seed(time.Now().Unix())
    canvas.Start(width, height)
    canvas.Rect(0,0,width,height)
    for i := 0; i < nstars; i++ {
        canvas.Circle(rn(width), rn(height), rn(3), "fill:white")
    }
    canvas.Circle(width/2, height, width/2, "fill:rgb(77, 117, 232)")
    canvas.Text(width/2, height*4/5, "hello, world", style)
    canvas.End()
}

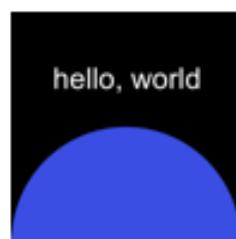
(Shift-Enter to compile and run.)  Compile and run after each keystroke
```



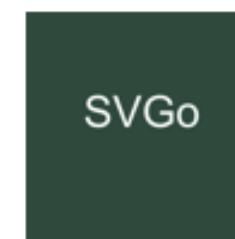
ajstarks@slab:~

```
$ svgplay
2019/06/18 19:19:44 ⚠️⚠️⚠️ Warning: this server allows a client connecting to 127.0.0.1:1999 to execute code on this computer ⚠️⚠️⚠️
```

## SVGo Examples



hello.go



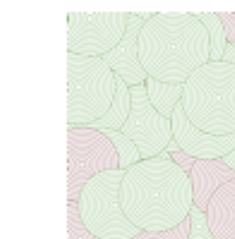
SVGo



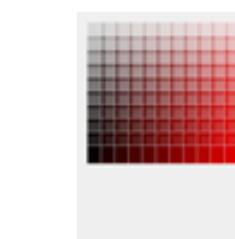
rl.go



concentric.go



concentric2.go



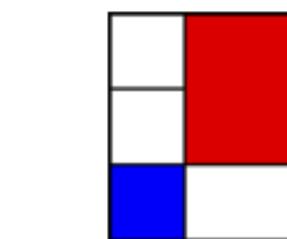
cgrid.go



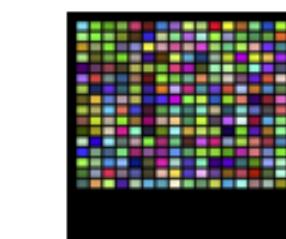
diag.go



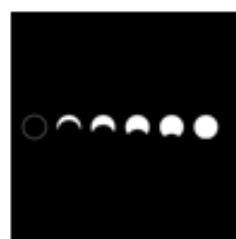
shining.go



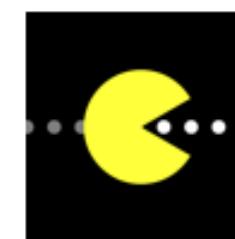
mondrian.go



richter.go



eclipse.go



pacman.go



pyramid.go



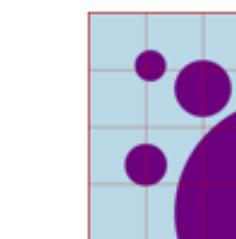
cloud.go



color-clouds.go



creepy.go



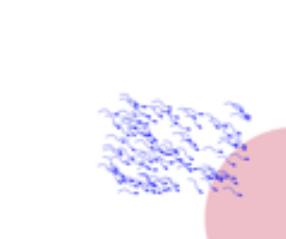
d4h.go



sunearth.go



conception.go



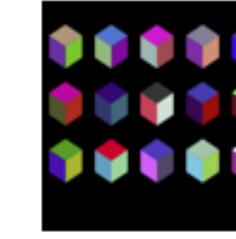
conception2.go



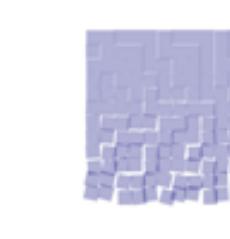
randbox.go



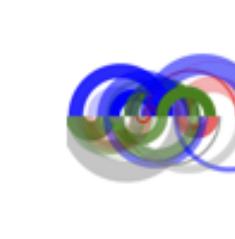
randspot.go



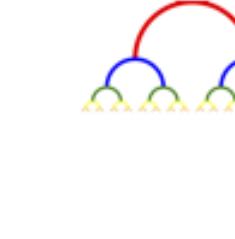
cube.go



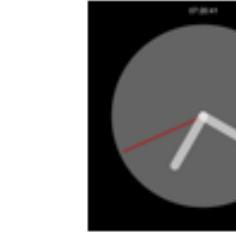
schotter.go



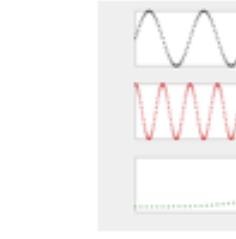
randarc.go



reurse.go



clock.go



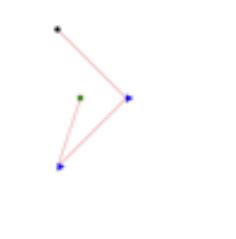
plotfunc.go



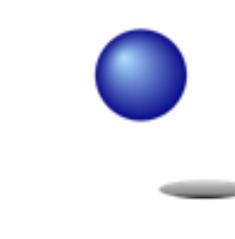
therm.go



pattern.go



marker.go



gradient.go



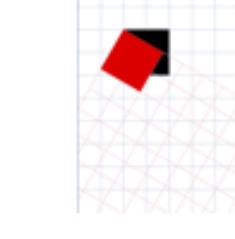
fontrange.go



uni.go



lorem.go



rotate.go



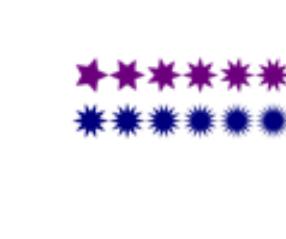
rottext.go



skewabc.go



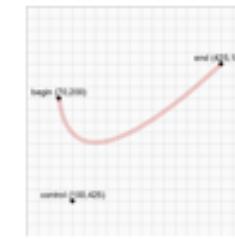
gear.go



star.go



starx.go



swoosh.go



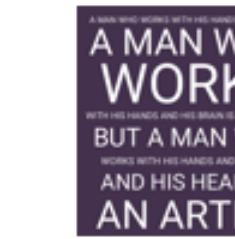
textpath.go



imfade.go



fe.go



artist.go



go.go

code/svgplay-samples

```

package main

import (
    "fmt"
    "github.com/ajstarks/svg"
    "os"
)

func main() {
    xp := []int{50, 70, 70, 50, 30, 30}
    yp := []int{40, 50, 75, 85, 75, 50}
    xl := []int{0, 0, 50, 100, 100}
    yl := []int{100, 40, 10, 40, 100}
    bgcolor := "rgb(227,78,25)"
    bkcolor := "rgb(153,29,40)"
    stcolor := "rgb(65,52,44)"
    stwidth := 12
    stylefmt := "stroke:%s;stroke-width:%d;fill:%s"
    canvas := svg.New(os.Stdout)
    width, height := 500, 500
    canvas.Start(width, height)
    canvas.Def()
    canvas.Gid("unit")
    canvas.Polyline(xl, yl, "fill:none")
    canvas.Polygon(xp, yp)
    canvas.Gend()
    canvas.Gid("runit")
    canvas.TranslateRotate(150, 180, 180)
    canvas.Use(0, 0, "#unit")
    canvas.Gend()
    canvas.Gend()
    canvas.DefEnd()
    canvas.Rect(0, 0, width, height, "fill:"+bgcolor)
    canvas.Gstyle(fmt.Sprintf(stylefmt, stcolor, stwidth, bkcolor))
    for y := 0; y < height; y += 130 {
        for x := -50; x < width; x += 100 {
            canvas.Use(x, y, "#unit")
            canvas.Use(x, y, "#runit")
        }
    }
    canvas.Gend()
    canvas.End()
}

```



shining.go [8]

```

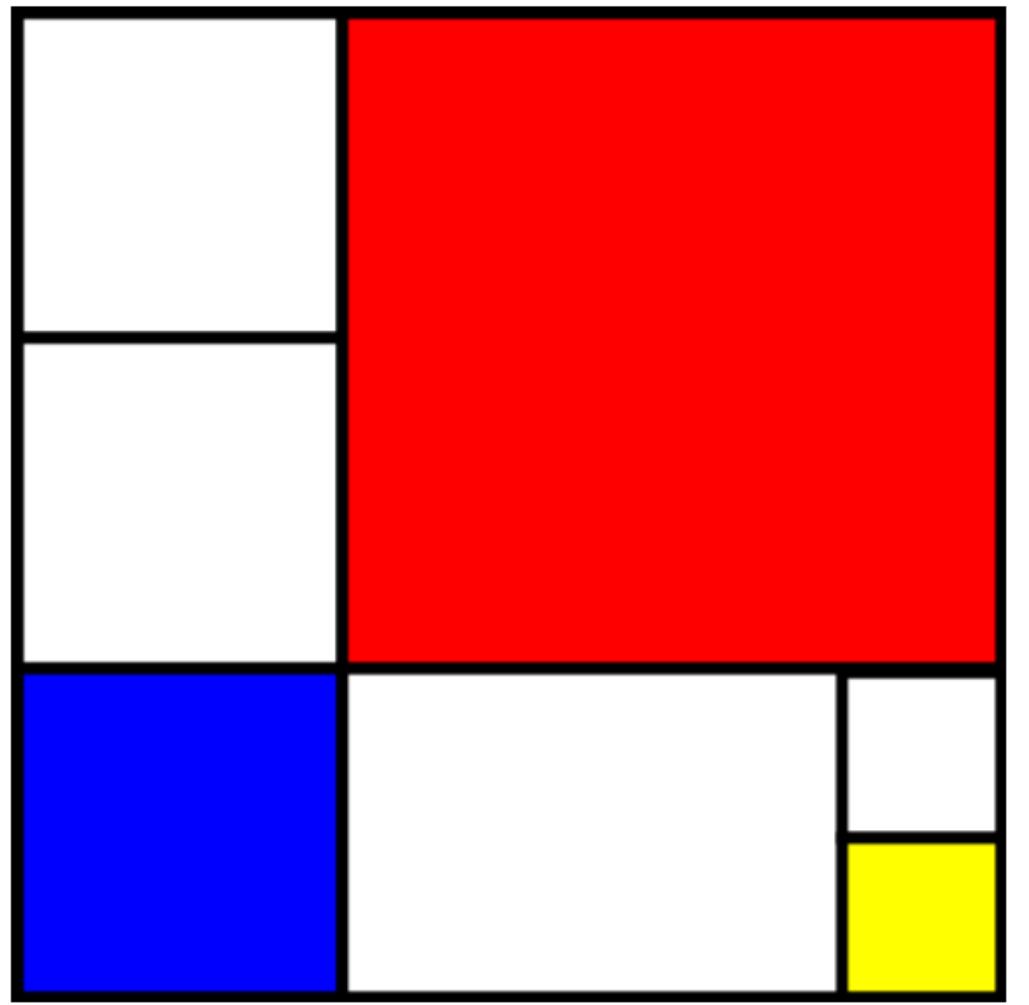
package main

import (
    "os"
    "github.com/ajstarks/svg"
)

var (
    canvas = svg.New(os.Stdout)
    width  = 500
    height = 500
)

// Piet Mondrian - Composition in Red, Blue, and Yellow
func main() {
    w3 := width / 3
    w6 := w3 / 2
    w23 := w3 * 2
    canvas.Start(width, height)
    canvas.Gstyle("stroke:black;stroke-width:6")
    canvas.Rect(0, 0, w3, w3, "fill:white")
    canvas.Rect(0, w3, w3, w3, "fill:white")
    canvas.Rect(0, w23, w3, w3, "fill:blue")
    canvas.Rect(w3, 0, w23, w23, "fill:red")
    canvas.Rect(w3, w23, w23, w3, "fill:white")
    canvas.Rect(width-w6, height-w3, w3-w6, w6, "fill:white")
    canvas.Rect(width-w6, height-w6, w3-w6, w6, "fill:yellow")
    canvas.Gend()
    canvas.Rect(0, 0, width, height, "fill:none;stroke:black;stroke-width:12")
    canvas.End()
}

```



mondrian.go [9]

```

package main

import (
    "github.com/ajstarks/svg"
    "os"
)

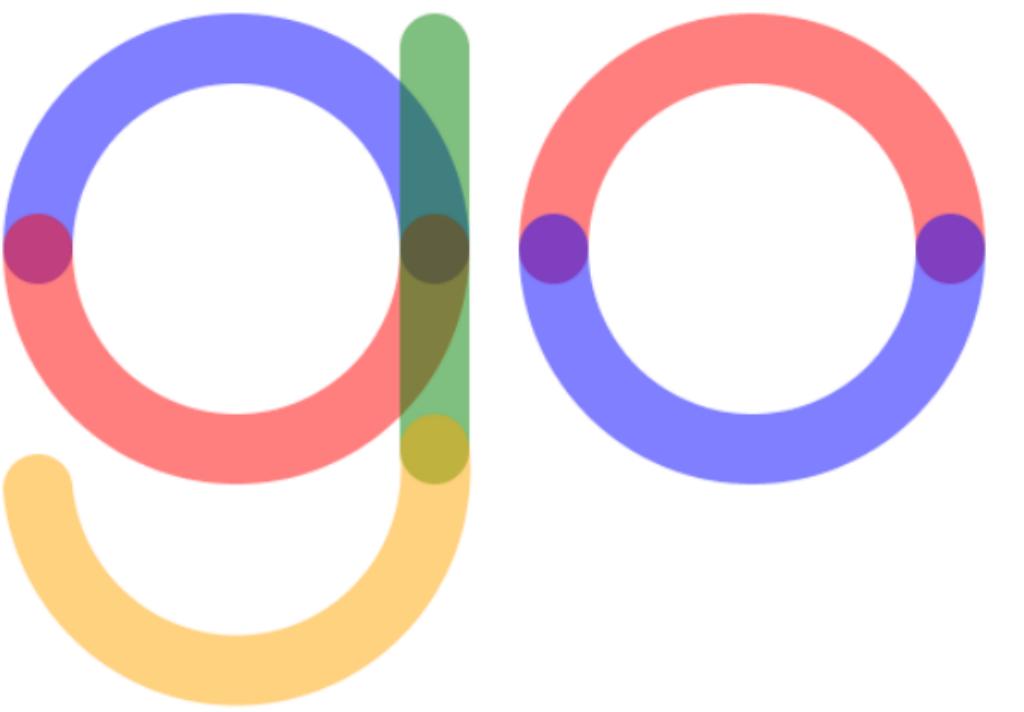
var (
    canvas = svg.New(os.Stdout)
    width  = 500
    height = 500
)

func main() {
    blues := "stroke:blue"
    reds := "stroke:red"
    greens := "stroke:green"
    organges := "stroke:orange"

    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:white")

    canvas.Gstyle("fill:none;stroke-opacity:0.5;stroke-width:35;stroke-linecap:round")
    // g
    canvas.Arc(20, 200, 30, 30, 0, false, true, 220, 200, blues)
    canvas.Arc(20, 200, 30, 30, 0, false, false, 220, 200, reds)
    canvas.Line(220, 100, 220, 300, greens)
    canvas.Arc(20, 320, 30, 30, 0, false, false, 220, 300, organges)
    // o
    canvas.Arc(280, 200, 30, 30, 0, false, true, 480, 200, reds)
    canvas.Arc(280, 200, 30, 30, 0, false, false, 480, 200, blues)
    canvas.Gend()
    canvas.End()
}

```



go.go [47]

```

package main

import (
    "github.com/ajstarks/svg"
    "os"
)

var (
    canvas = svg.New(os.Stdout)
    width  = 500
    height = 500
)

func main() {
    angle, cx, cy := 30.0, width/2, height/2
    r := width / 4
    p := r / 8

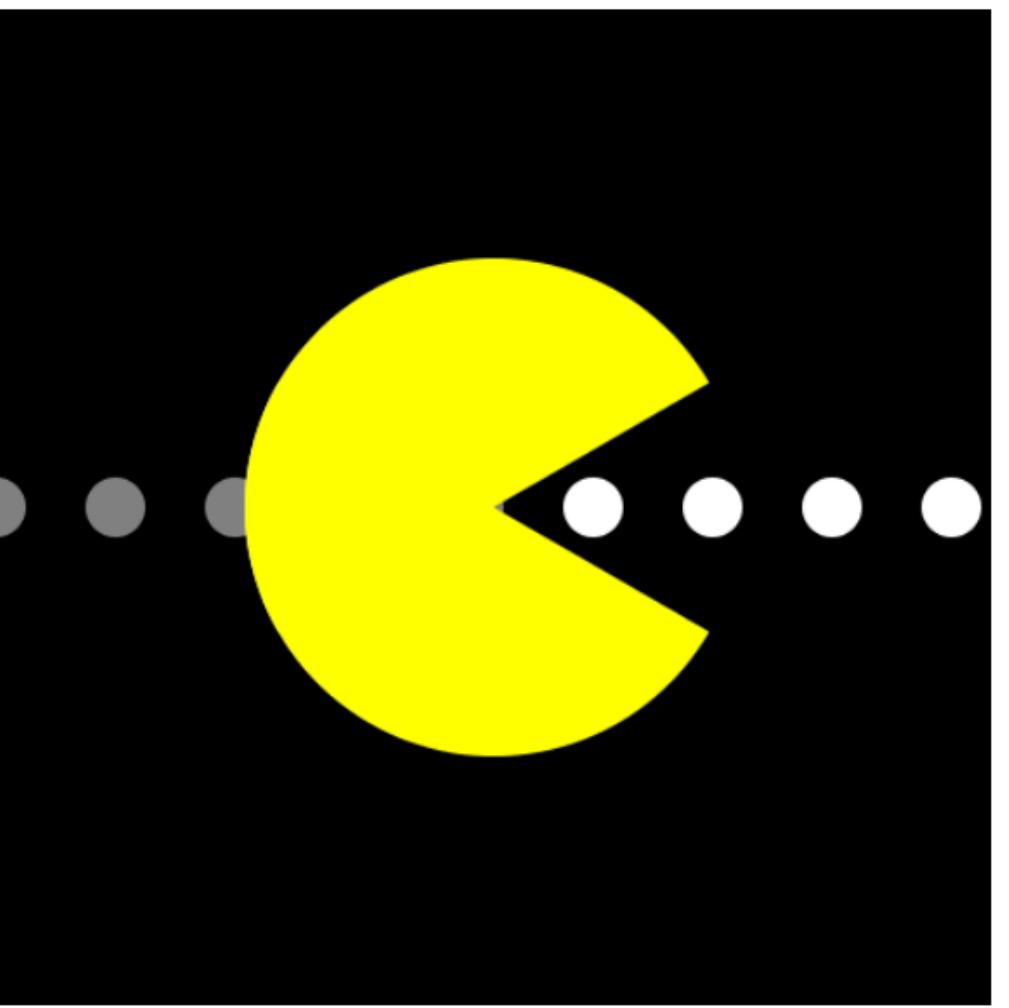
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Gstyle("fill:white")
    for x := 0; x < 100; x += 12 {
        if x < 50 {
            canvas.Circle((width*x)/100, cy, p, "fill-opacity:0.5")
        } else {
            canvas.Circle((width*x)/100, cy, p, "fill-opacity:1")
        }
    }
    canvas.Gend()

    canvas.Gstyle("fill:yellow")
    canvas.TranslateRotate(cx, cy, -angle)
    canvas.Arc(-r, 0, r, r, 30, false, true, r, 0)
    canvas.Gend()

    canvas.TranslateRotate(cx, cy, angle)
    canvas.Arc(-r, 0, r, r, 30, false, false, r, 0)
    canvas.Gend()

    canvas.Gend()
    canvas.End()
}

```



pacman.go [12]

# Read/Parse/Draw Pattern

# Read

# Parse

# Draw

from the data source    into data structures    from the structures

# Data



# Picture

# Read

<https://api.nytimes.com/svc/topstories/v2/food.json?api-key=...>

```
{  
  "status": "OK",  
  "copyright": "Copyright (c) 2019 The New York Times Company. All Rights Reserved.",  
  "section": "food",  
  "last_updated": "2019-06-26T07:57:20-04:00",  
  "num_results": 26,  
  "results": [  
    {  
      "section": "Food",  
      "subsection": "",  
      "title": "Thomas Keller Brings Country Club Cuisine to the City",  
      "abstract": "At TAK Room at Hudson Yards, the chef salutes the strait-laced, spice-free food that rich Americans used to feed on.",  
      "url": "https://www.nytimes.com/2019/06/25/dining/tak-room-review-thomas-keller.html",  
      "byline": "By PETE WELLS",  
      "item_type": "Article",  
      "updated_date": "2019-06-25T11:59:42-04:00",  
      "created_date": "2019-06-25T11:59:42-04:00",  
      "published_date": "2019-06-25T11:59:42-04:00",  
      "material_type_facet": "",  
      "kicker": "",  
      "des_facet": [  
        "Restaurants"  
      ],  
      "org_facet": [  
        "TAK Room (Manhattan, NY, restaurant)"  
      ],  
      "per_facet": [  
        "Keller, Thomas"  
      ],  
      "geo_facet": [  
        "Hudson Yards (Manhattan, NY)"  
      ],  
      "multimedia": [  
        {  
          "url": "https://static01.nyt.com/images/2019/06/26/dining/25REST-slide-H6ZN/25REST-slide-H6ZN-thumbStandard.jpg",  
          "format": "Standard Thumbnail",  
          "height": 75,  
          "width": 75,  
          "type": "image",  
          "subtype": "photo",  
          "caption": "",  
          "copyright": "Jeenah Moon for The New York Times"  
        },  
        {  
          "url": "https://static01.nyt.com/images/2019/06/26/dining/25REST-slide-H6ZN/25REST-slide-H6ZN-thumbLarge.jpg",  
          "format": "thumbLarge",  
          "height": 150.  
        }  
      ]  
    }  
  ]  
}
```

# Parse

# Draw



# Imports

```
package main

import (
    "encoding/json"
    "flag"
    "fmt"
    "io"
    "net/http"
    "os"
    "time"

    "github.com/ajstarks/svgो"
)
```

# Constants

```
// API Info and formats
const (
    NYTAPIkey = "HxTV50RYD7LyGzHxUy7XCThV3kQ3HYvk"
    NYTfmt    = "https://api.nytimes.com/svc/topstories/v2/%s.json?api-key=%s"
    style     = "font-size:20pt;font-family:sans-serif;fill:white"
    errfmt    = "unable to get network data for %s (%s)"
    headfmt   = "New York Times Top Stories for %s"
    datefmt   = "Monday Jan 2, 2006"
    usage     = `section choices:
arts, automobiles, books, business, fashion,
food, health, home, insider, magazine, movies,
national, nyregion, obituaries, opinion, politics,
realestate, science, sports, sundayreview, technology,
theater, tmagazine, travel, upshot, world`)
```

# Data Structures

```
// NYTStories is the headline info from the New York Times
type NYTStories struct {
    StoryCount int      `json:"num_results"`
    Results     []result `json:"results"`
}

type result struct {
    Title      string   `json:"title"`
    URL       string   `json:"url"`
    Multimedia []multimedia `json:"multimedia"`
}

type multimedia struct {
    URL      string `json:"url"`
    Width   int    `json:"width"`
    Height  int    `json:"height"`
    Format  string `json:"format"`
}
```

# Main

```
func main() {
    var section = flag.String("s", "home", usage)
    var nstories = flag.Int("n", 25, "number of stories")
    flag.Parse()

    width, height := 1000, (*nstories/5)*150
    canvas := svg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    nytStories(canvas, width, *section, *nstories)
    canvas.End()
}
```

# Network

```
// netread dereferences a URL, returning the Reader, with an error
func netread(url string) (io.ReadCloser, error) {
    client := &http.Client{Timeout: 30 * time.Second}
    resp, err := client.Get(url)
    if err != nil {
        return nil, err
    }
    if resp.StatusCode != http.StatusOK {
        return nil, fmt.Errorf(errfmt, url, resp.Status)
    }
    return resp.Body, nil
}
```

# Read and Decode

```
// nytStories retrieves Top Stories data
// from the New York Times API, decodes and displays it.
func nytStories(canvas *svg.SVG, width int, section string, n int) {
    r, err := netread(fmt.Sprintf(NYTfmt, section, NYTAPIkey))
    if err != nil {
        fmt.Fprintf(os.Stderr, "headline read error: %v\n", err)
        return
    }
    defer r.Close()
    var data NYTStories
    if err = json.NewDecoder(r).Decode(&data); err != nil {
        fmt.Fprintf(os.Stderr, "decode: %v\n", err)
        return
    }
    drawStories(canvas, width, data, n)
}
```

## Drawing (setup)

```
// drawStories walks the result structure,  
// displaying title and thumbnail in a grid  
func drawStories(canvas *svg.SVG, width int, data NYTStories, n int) {  
    top, left := 100, 150  
    titley := top - 50  
    x, y := left, top  
    ts := fmt.Sprintf(headfmt, time.Now().Format(datefmt))  
    if n > data.StoryCount {  
        n = data.StoryCount  
    }
```

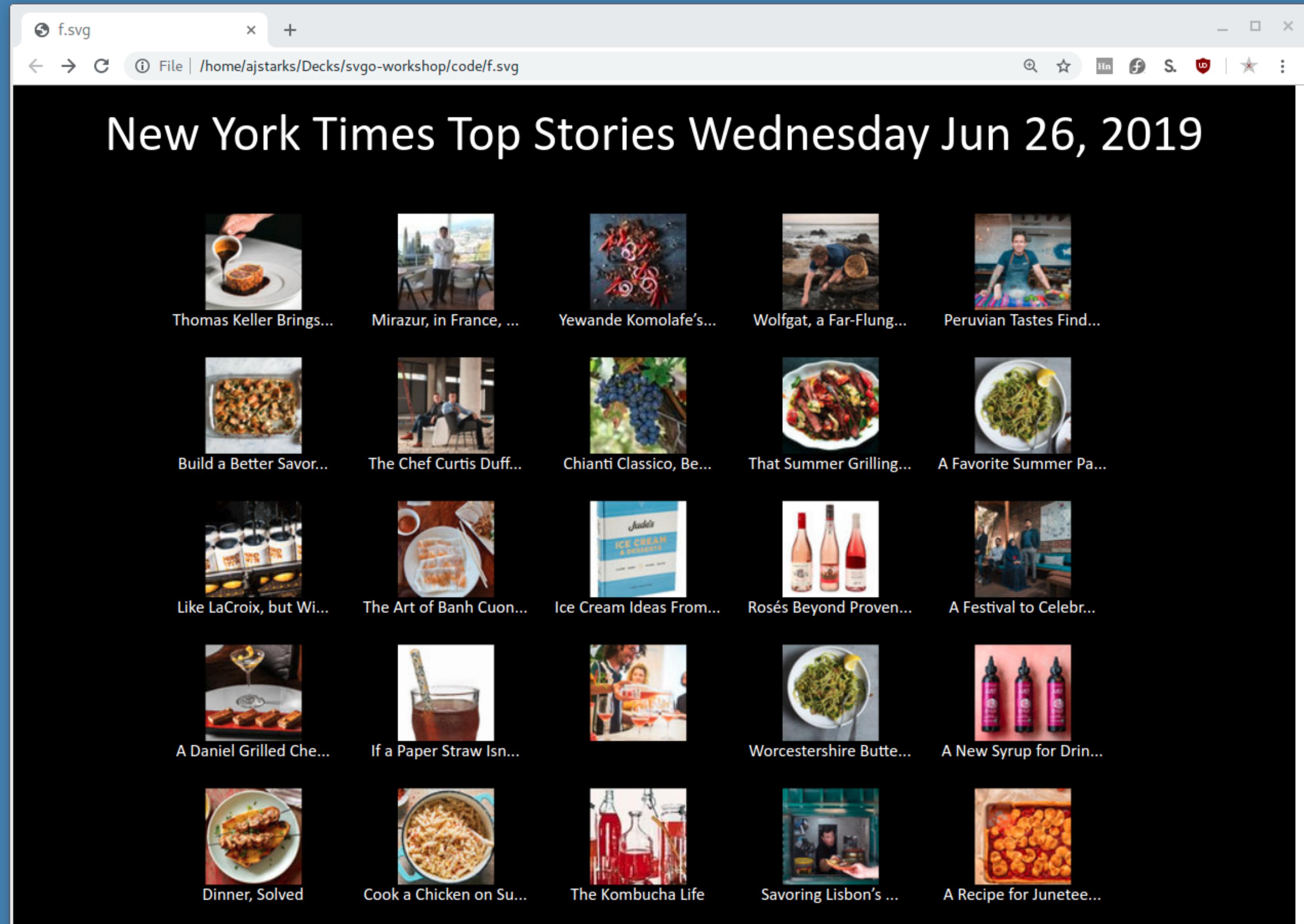
# Drawing

```
canvas.Gstyle(style)
canvas.Text(width/2, titley, ts, "font-size:300%")
for i := 0; i < n; i++ {
    d := data.Results[i]
    tw, th, imagelink := imageinfo(d)
    if i > 0 && i%5 == 0 {
        x = left
        y += th + (th / 2)
    }
    canvas.Link(d.URL, d.Title)
    canvas.Image(x, y, tw, th, imagelink)
    canvas.Text(x+tw/2, y+th+12, struncate(d.Title, 20))
    canvas.LinkEnd()
    x += tw * 2
}
canvas.Gend()
```

# Helpers

```
// struncate truncates a string with ellipsis
func struncate(s string, n int) string {
    if len(s) <= n {
        return s
    }
    return s[0:n] + "..."
}

// imageinfo returns the thumbnail image information
func imageinfo(data result) (int, int, string) {
    for _, m := range data.Multimedia {
        if m.Format == "Standard Thumbnail" {
            return m.Width, m.Height, m.URL
        }
    }
    return 75, 75, ""
}
```



🔍

ajstarks@slab:~/Decks/svgo-workshop/code

```
$ go run nytsvg.go -s food > f.svg
```



# Thank You

<https://github.com/ajstarks/svgo-workshop>

Anthony Starks  
@ajstarks