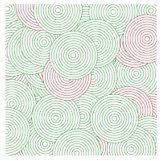
gensvg examples

hello

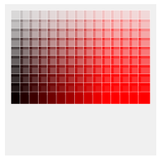
colorhash

rl

concentric



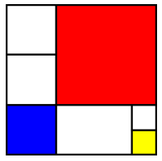
concentric2



cgrid

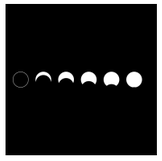
diag

shining



mondrian

richter



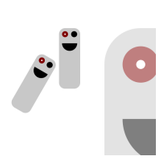
eclipse

pacman

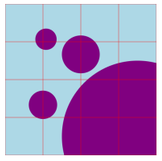
pyramid

cloud

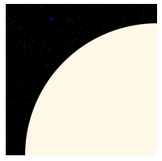
color-clouds



creepy

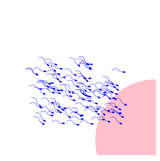


d4h



sunearth

conception

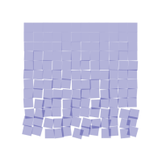


conception2

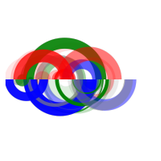
randbox

randspot

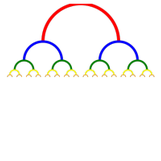
cube



schotter

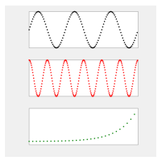


randarc



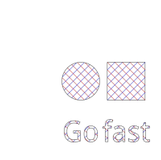
recurse

clock

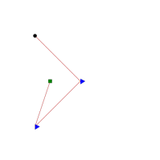


plotfunc

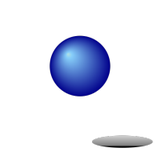
therm



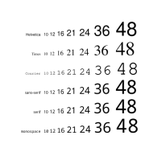
pattern



marker



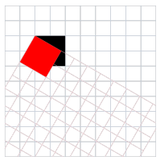
gradient



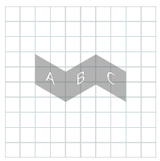
fontrange

uni

lorem



rotate



skewabc

gear

star

starx

swoosh

textpath

imfade

fe

artist

go

am

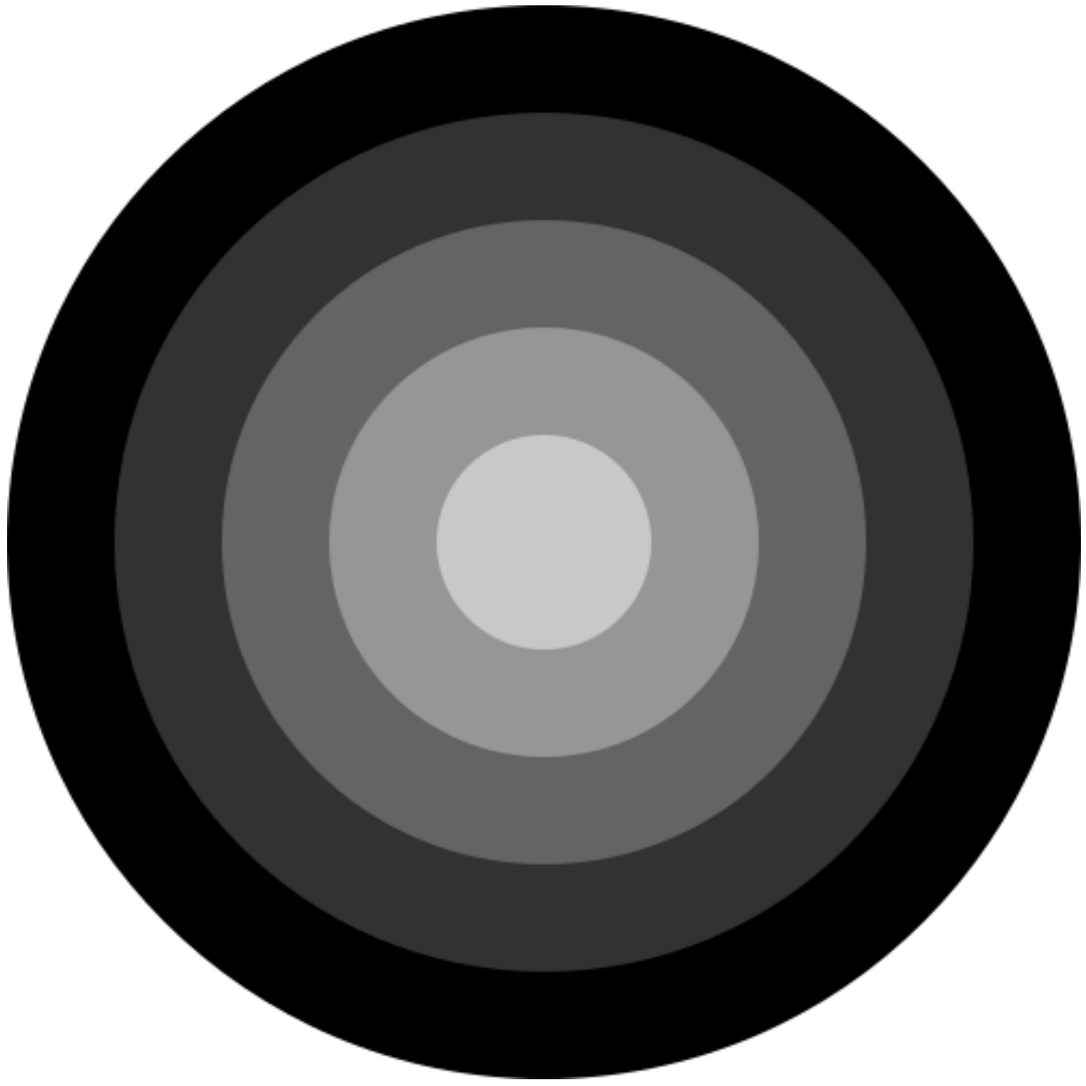
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    style := "fill:white;font-size:48pt;text-anchor:middle"
    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/3, "hello, world", style)
    canvas.End()
}
```

hello.go [1]

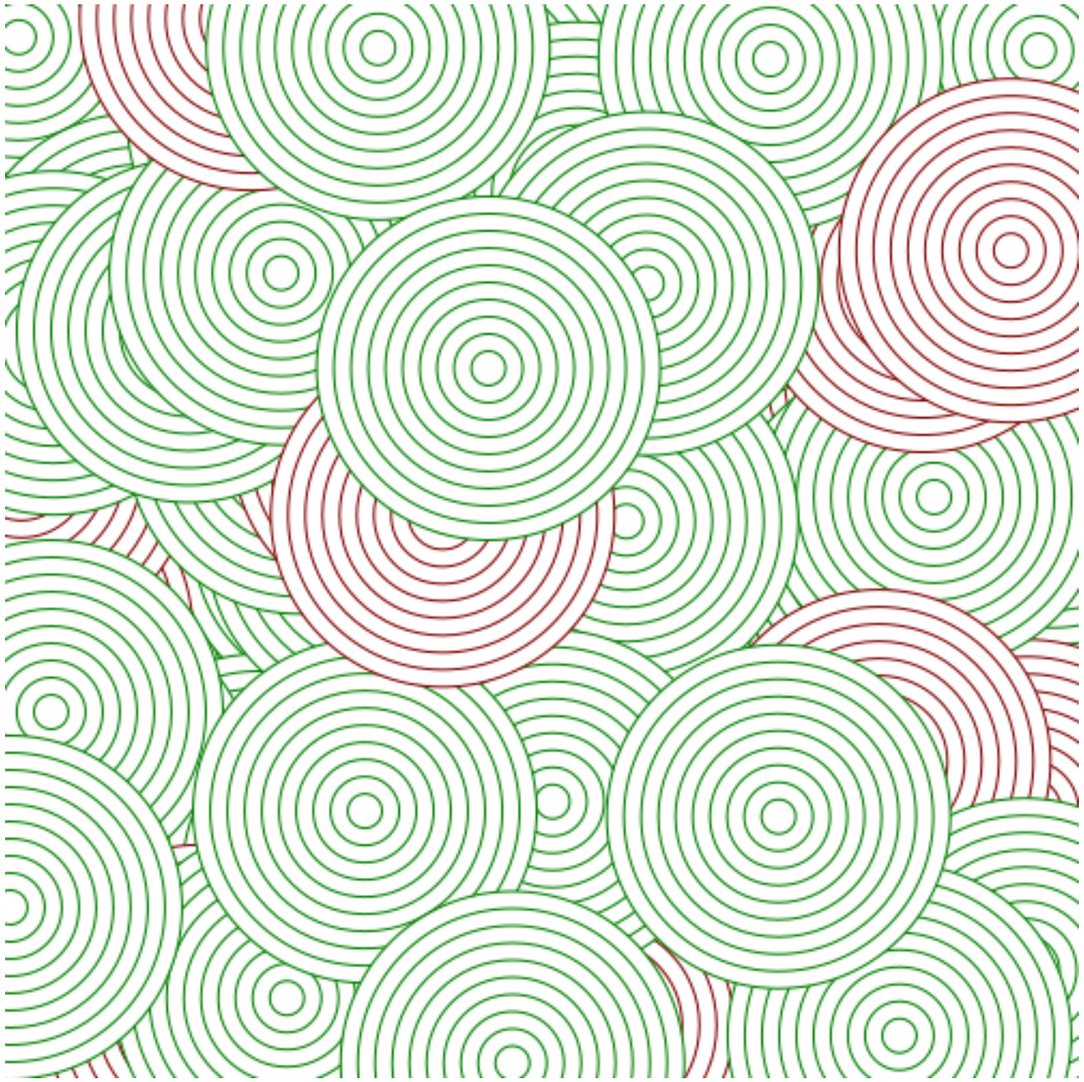
```
package main

import (
    "crypto/md5"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func colorhash(s string) (int, int, int) {
    hash := md5.New()
    hash.Write([]byte(s))
    v := hash.Sum(nil)
    return int(v[0]), int(v[1]), int(v[2])
}

func main() {
    name := "SVGo"
    style := "fill:white;text-anchor:middle;font-size:72pt"
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, canvas.RGB(colorhash(name)))
    canvas.Text(width/2, height/2, name, style)
    canvas.End()
}
```

```
package main

import (
    "fmt"
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    rand.Seed(time.Now().Unix())
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Gstyle("stroke-width:10")

    for i := 0.0; i < width; i++ {
        r := rand.Intn(255)
        canvas.Line(i, 0, rand.Float64()*width, height,
            fmt.Sprintf("stroke:rgb(%d,%d,%d); opacity:0.39", r, r, r))
    }
    canvas.Gend()
    canvas.End()
}
```

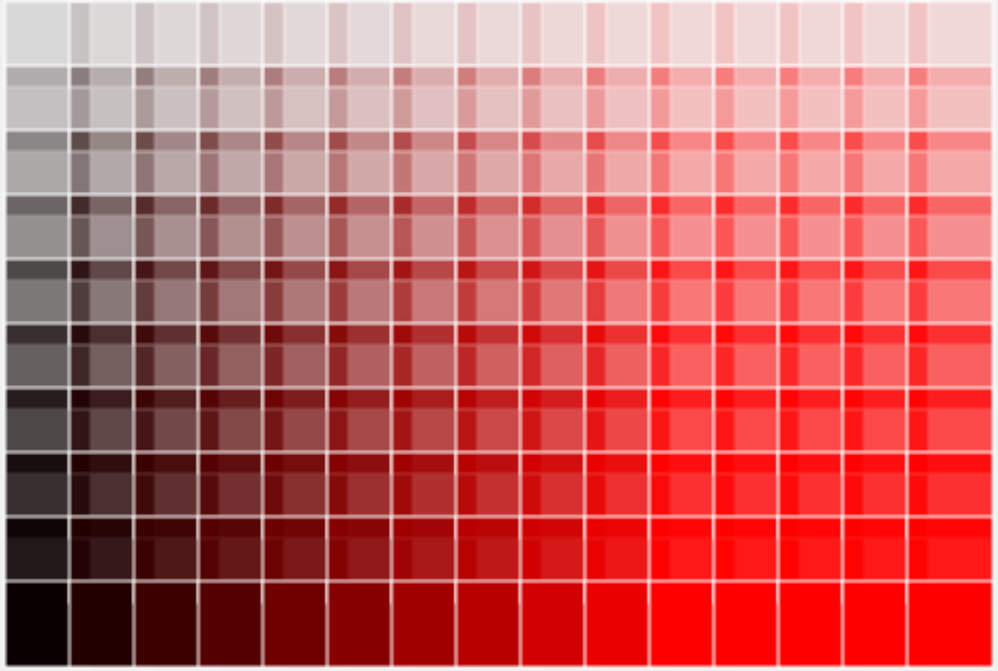
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:white")
    r := height / 2
    for g := 0; g < 250; g += 50 {
        canvas.Circle(width/2, width/2, r, canvas.RGB(g, g, g))
        r -= 50
    }
    canvas.End()
}
```



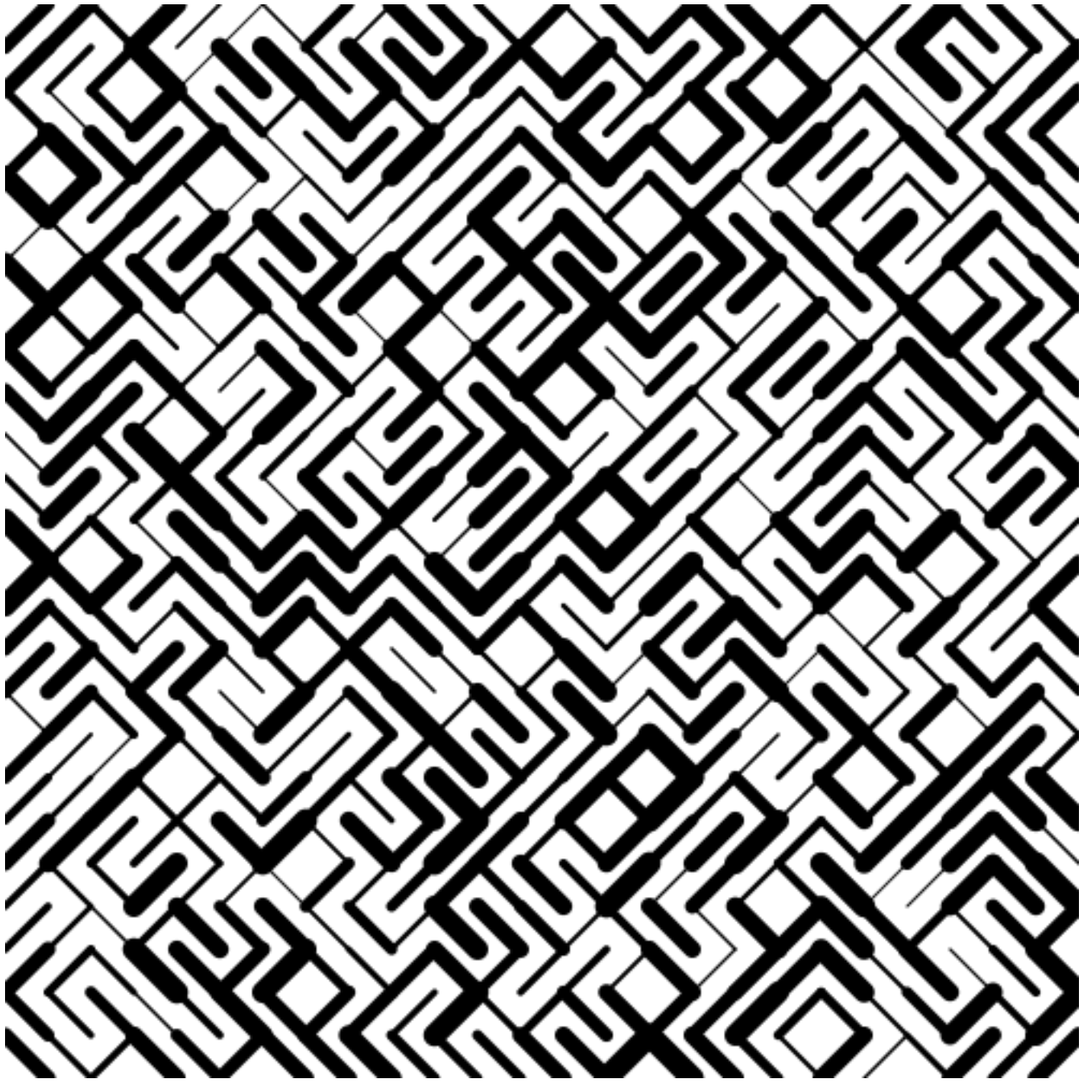
```
package main

import (
    "math/rand"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    canvas.Start(width, height)
    canvas.Gstyle("fill:white")
    var color string
    radius := 80.0
    step := 8.0
    for i := 0; i < 200; i++ {
        if i%4 == 0 {
            color = "rgb(127,0,0)"
        } else {
            color = "rgb(0,127,0)"
        }
        x, y := rand.Float64()*(width), rand.Float64()*(height)
        for r, nc := radius, 0; nc < 10; nc++ {
            canvas.Circle(x, y, r, "stroke:"+color)
            r -= step
        }
    }
    canvas.Gend()
    canvas.End()
}
```



```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    y := 20.0
    v := 10
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:rgb(240,240,240)")
    canvas.Gstyle("stroke:white")
    for x := 20.0; x < 450; x += 30 {
        op := 0.1
        for i := 0; i < 100; i += 10 {
            canvas.Square(x, y, 20*2, canvas.RGBA(v, 0, 0, op))
            y += 30
            op += 0.1
        }
        y = 20
        v += 25
    }
    canvas.Gend()
    canvas.End()
}
```

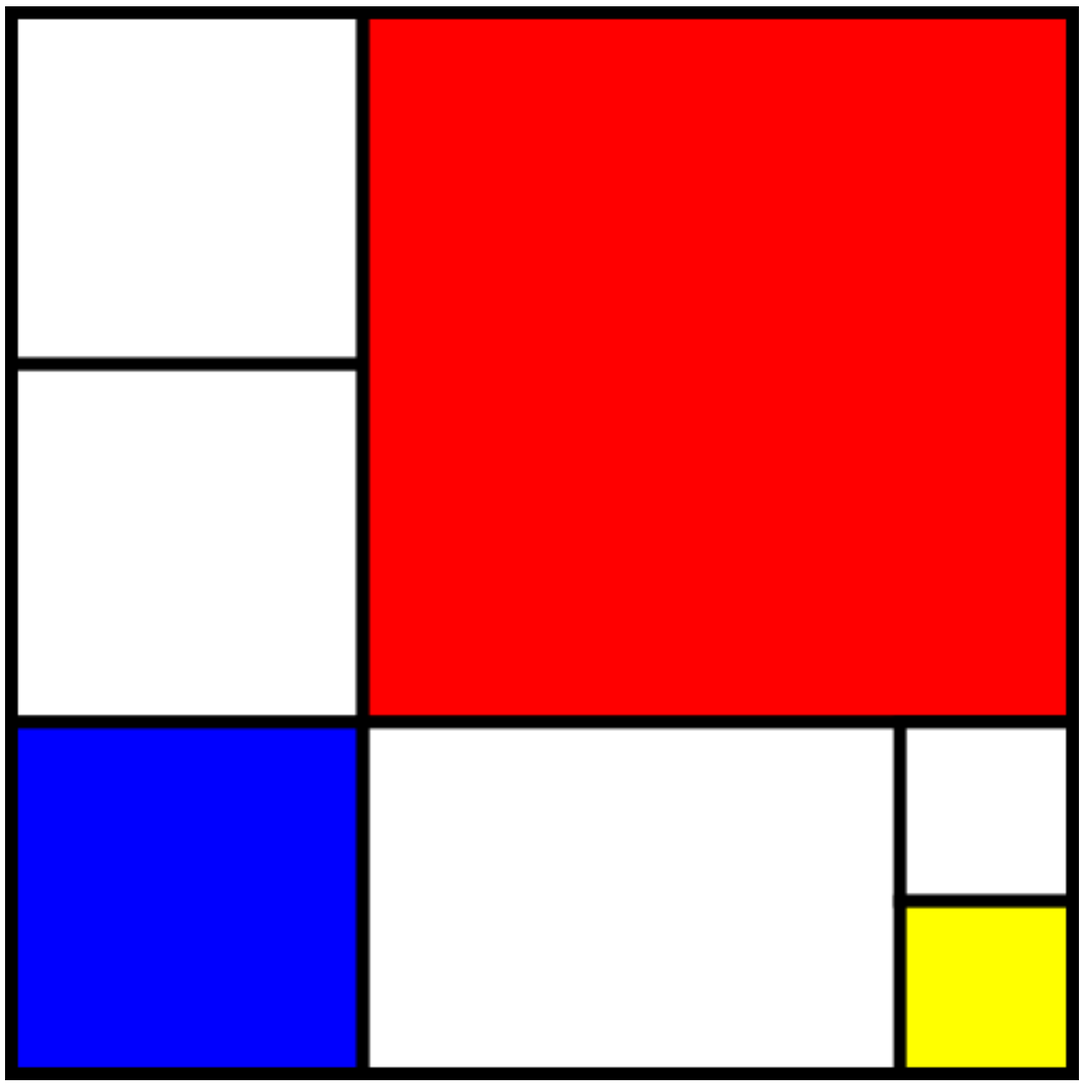


```
package main

import (
    "fmt"
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    tiles, maxstroke := 25.0, 10.0
    rand.Seed(time.Now().Unix())
    canvas.Start(width, height)
    linecaps := []string{"butt", "round", "square"}
    strokefmt := "stroke-width:%.2f"
    lcfmt := "stroke:black;stroke-linecap:%s"
    canvas.Gstyle(fmt.Sprintf(lcfmt, linecaps[rand.Intn(3)]))
    var sw string
    for y := 0.0; y < tiles; y++ {
        for x := 0.0; x < tiles; x++ {
            px := width / tiles * x
            py := height / tiles * y
            if rand.Intn(100) > 50 {
                sw = fmt.Sprintf(strokefmt, rand.Float64()*(maxstroke)+1)
                canvas.Line(px, py, px+width/tiles, py+height/tiles, sw)
            } else {
                sw = fmt.Sprintf(strokefmt, rand.Float64()*(maxstroke)+1)
                canvas.Line(px, py+height/tiles, px+width/tiles, py, sw)
            }
        }
    }
    canvas.Gend()
    canvas.End()
}
```



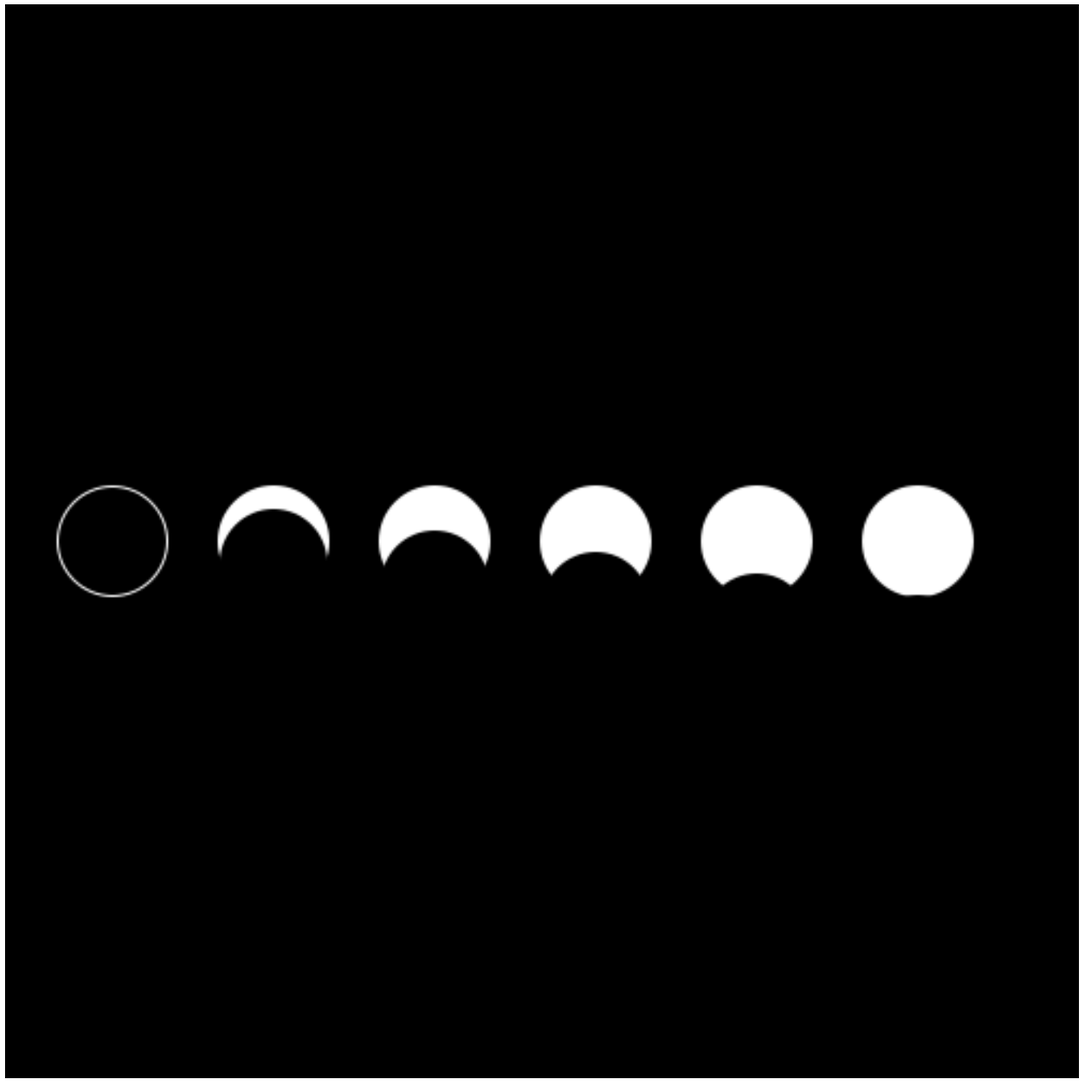
```
package main

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

func main() {
    xp := []float64{50, 70, 70, 50, 30, 30}
    yp := []float64{40, 50, 75, 85, 75, 50}
    xl := []float64{0, 0, 50, 100, 100}
    yl := []float64{100, 40, 10, 40, 100}
    bgcolor := "rgb(227,78,25)"
    bkcolor := "rgb(153,29,40)"
    stcolor := "rgb(65,52,44)"
    stwidth := 12.0
    stylefmt := "stroke:%s;stroke-width:%.2f;fill:%s"
    canvas := gensvg.New(os.Stdout)
    width, height := 500.0, 500.0
    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 := 0.0; y < height; y += 130 {
        for x := -50.0; x < width; x += 100 {
            canvas.Use(x, y, "#unit")
            canvas.Use(x, y, "#runit")
        }
    }
    canvas.Gend()
    canvas.End()
}
```

```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

// 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()
}
```

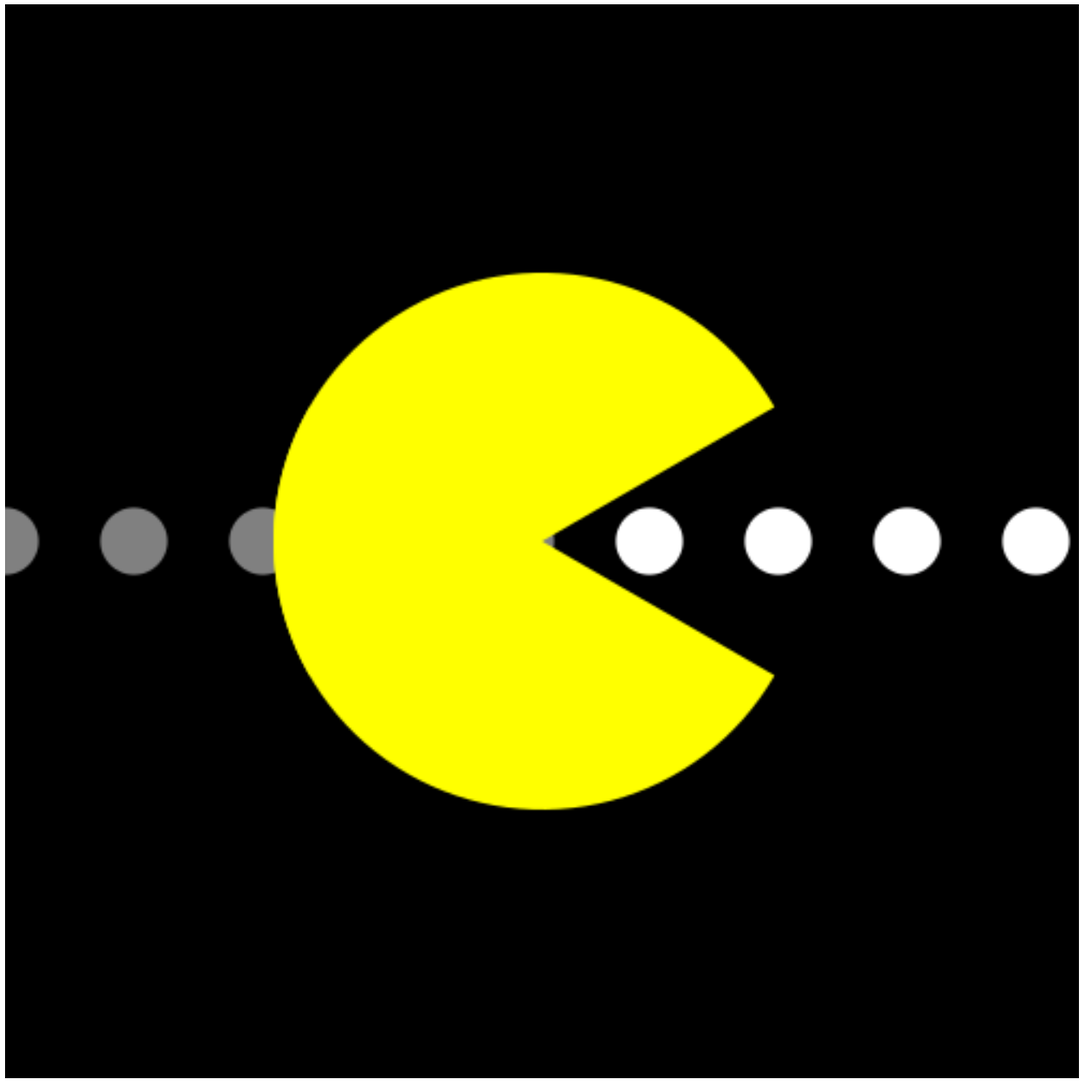


```
package main

import (
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

// inspired by Gerhard Richter's 256 colors, 1974
func main() {
    rand.Seed(time.Now().Unix())
    canvas.Decimals=0
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)

    w, h, gutter := 24.0, 18.0, 5.0
    rows, cols := 16.0, 16.0
    top, left := 20.0, 20.0

    for r, x := 0.0, left; r < rows; r++ {
        for c, y := 0.0, top; c < cols; c++ {
            canvas.Rect(x, y, w, h,
                canvas.RGB(rand.Intn(255), rand.Intn(255), rand.Intn(255)))
            y += (h + gutter)
        }
        x += (w + gutter)
    }
    canvas.End()
}
```

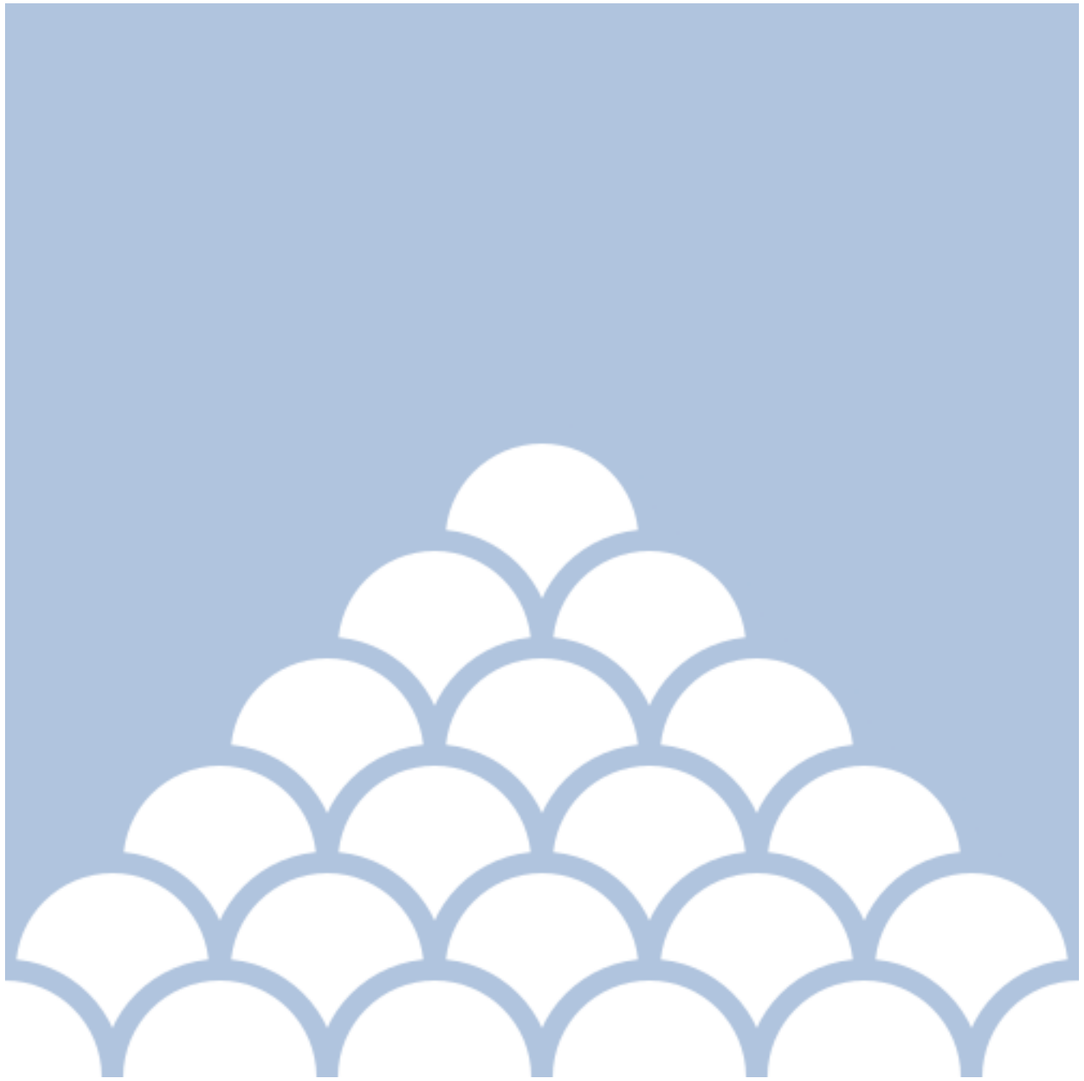
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    h2 := height / 2
    r := width / 20
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    for x, y := 50.0, h2; x < 450; x += 75 {
        canvas.Circle(x, h2, r+1, "fill:white")
        canvas.Circle(x, y, r, "fill:black")
        y += 10
    }
    canvas.End()
}
```



```
package main

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

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

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 := 0.0; 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()
}
```



```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    nr := 6
    radius := width / 10
    x := width / 2
    y := height / 2
    fgcolor := "white"
    bgcolor := "lightsteelblue"
    sw := width / 50
    sfmt := "fill:%s;;stroke:%s;stroke-width:%.2fpx"

    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:"+bgcolor)
    canvas.Gstyle(fmt.Sprintf(sfmt, fgcolor, bgcolor, sw))
    for r := 0; r < nr; r++ {
        xc := x
        for c := 0; c < r+1; c++ {
            canvas.Circle(xc, y, radius)
            xc += radius * 2
        }
        x -= radius
        y += radius
    }
    canvas.Gend()
    canvas.End()
}
```

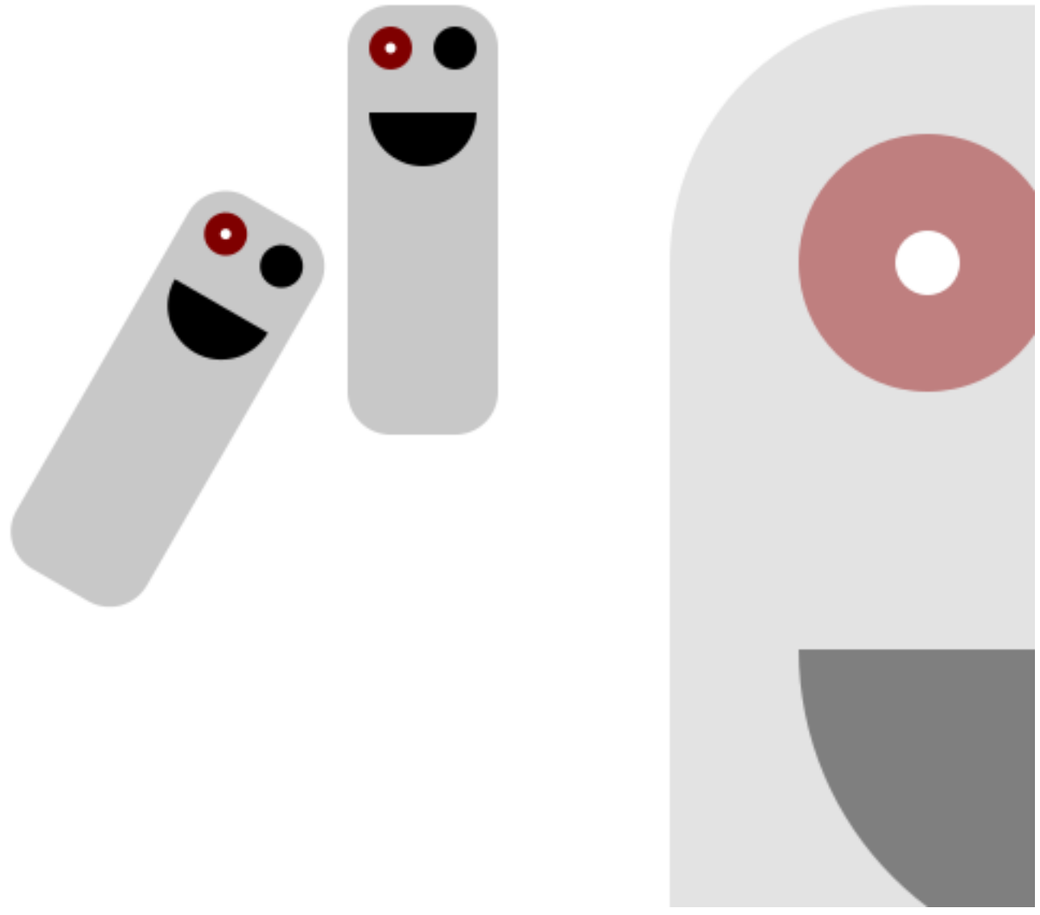


```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func cloud(x, y, r float64, style string) {
    small := r / 2
    medium := (r * 6) / 10
    canvas.Gstyle(style)
    canvas.Circle(x, y, r)
    canvas.Circle(x+r, y+small, small)
    canvas.Circle(x-r-small, y+small, small)
    canvas.Circle(x-r, y, medium)
    canvas.Rect(x-r-small, y, r*2+small, r)
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    cloud(width/2, height/2, 100, canvas.RGB(127, 127, 127))
    canvas.End()
}
```

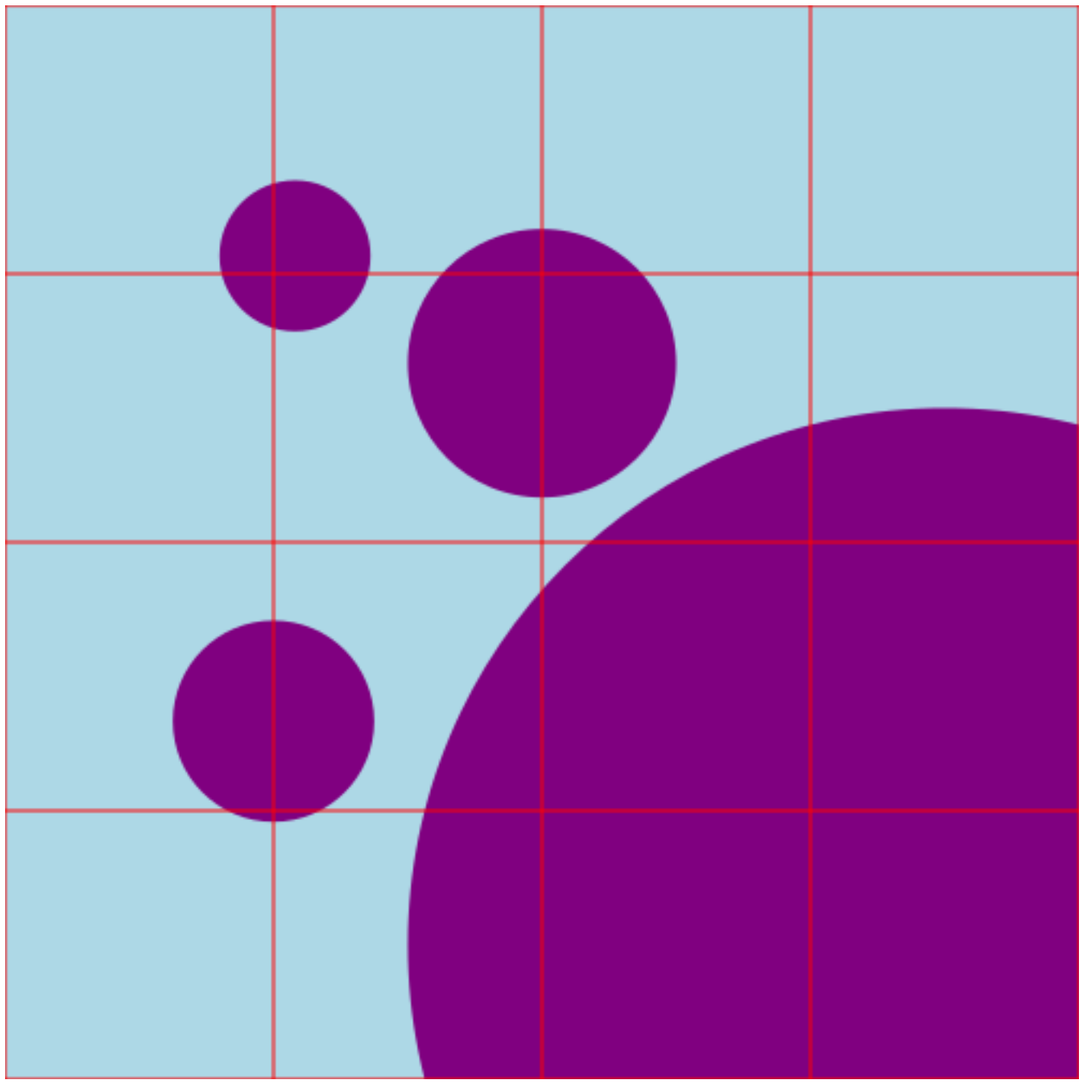
```
package main

import (
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func cloud(x, y, r float64, style string) {
    small := r / 2
    medium := (r * 6) / 10
    canvas.Gstyle(style)
    canvas.Circle(x, y, r)
    canvas.Circle(x+r, y+small, small)
    canvas.Circle(x-r-small, y+small, small)
    canvas.Circle(x-r, y, medium)
    canvas.Rect(x-r-small, y, r*2+small, r)
    canvas.Gend()
}

func main() {
    rand.Seed(time.Now().Unix())
    canvas.Start(width, height)
    for i := 0; i < 50; i++ {
        red := rand.Intn(255)
        green := rand.Intn(255)
        blue := rand.Intn(255)
        size := rand.Float64()*60
        x := rand.Float64()*width
        y := rand.Float64()*height
        cloud(x, y, size, canvas.RGB(red, green, blue))
    }
    canvas.End()
}
```

color-clouds.go [15]

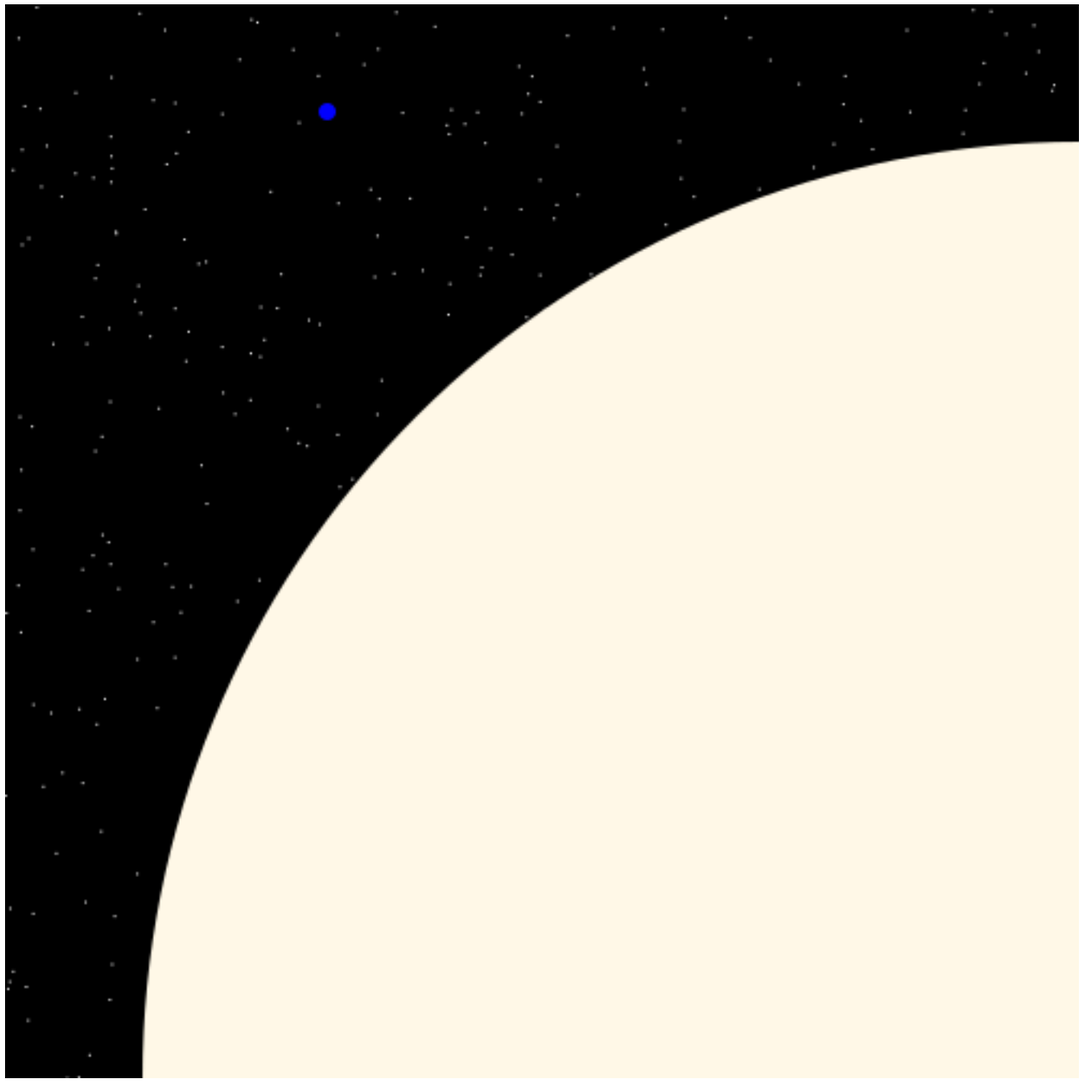
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var canvas = gensvg.New(os.Stdout)

func smile(x, y, r float64) {
    r2 := r * 2
    r3 := r * 3
    r4 := r * 4
    rq := r / 4
    gray := canvas.RGB(200, 200, 200)
    red := canvas.RGB(127, 0, 0)
    canvas.Roundrect(x-r2, y-r2, r*7, r*20, r2, r2, gray)
    canvas.Circle(x, y, r, red)
    canvas.Circle(x, y, rq, "fill:white")
    canvas.Circle(x+r3, y, r)
    canvas.Arc(x-r, y+r3, rq, rq, 0, true, false, x+r4, y+r3)
}

func main() {
    canvas.Start(500.0, 500)
    canvas.Rect(0, 0, 500.0, 500, "fill:white")
    smile(200, 100, 10)
    canvas.Gtransform("rotate(30)")
    smile(200, 100, 10)
    canvas.Gend()
    canvas.Gtransform("translate(50,0) scale(2,2)")
    canvas.Gstyle("opacity:0.5")
    smile(200, 100, 30)
    canvas.Gend()
    canvas.Gend()
    canvas.End()
}
```



```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func dot(x, y, d float64) {
    canvas.Circle(x, y, d/2, "fill:rgb(128,0,128)")
}

// Composition from "Design for Hackers, pg. 129
func main() {
    d1 := height
    d2 := d1 / 4
    d3 := (d2 * 3) / 4
    d4 := (d3 * 3) / 4

    coffset := height / 8
    hoffset := height / (height / 10)
    voffset := -width / 10

    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:lightblue")
    dot(width-coffset, height-coffset, d1)
    dot(width/2, height/3, d2)
    dot(width/4, height*2/3, d3)
    dot(width/4+hoffset, height/3+voffset, d4)
    canvas.Grid(0, 0, width, height, width/4, "stroke:red")
    canvas.End()
}
```

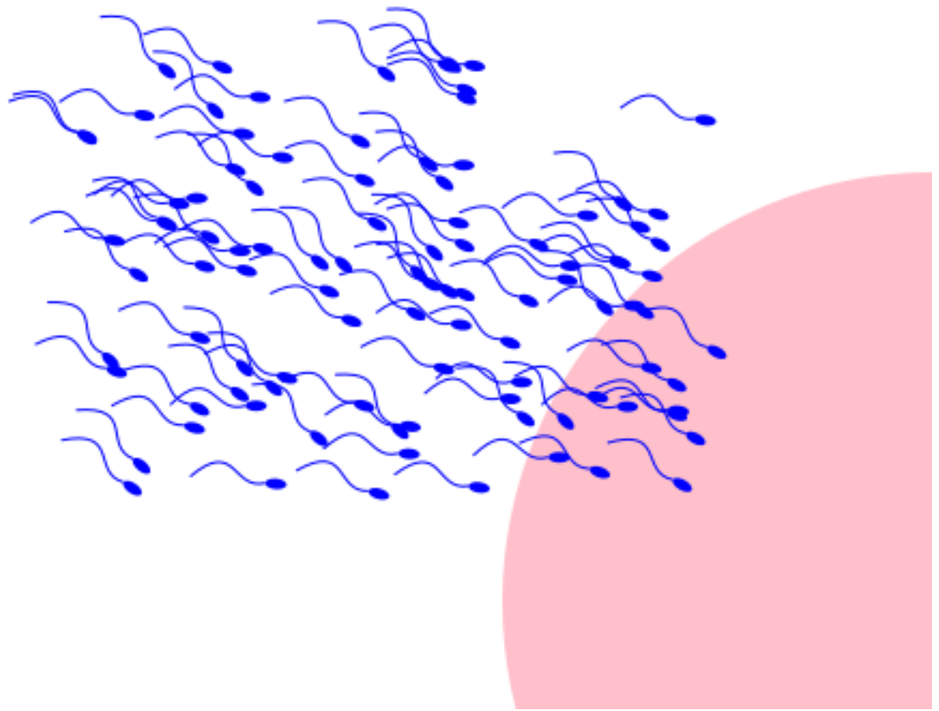


```
package main

import (
    "math/rand"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:black")
    for i := 0.0; i < width; i++ {
        x := rand.Float64()*width
        y := rand.Float64()*height
        canvas.Line(x, y, x, y+1, "stroke:white")
    }
    earth := 4.0
    sun := earth * 109
    canvas.Circle(150, 50, earth, "fill:blue")
    canvas.Circle(width, height, sun, "fill:rgb(255, 248, 231)")
    canvas.End()
}
```



```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func male(x, y, w float64) {
    canvas.Ellipse(x, y, w, w/2, "fill:blue")
    canvas.Bezier(
        x-(w*8), y,
        x-(w*4), y-(w*4),
        x-(w*4), y+w,
        x-w, y, "stroke:blue;fill:none")
}

func female(x, y, w float64) {
    canvas.Circle(x, y, w, "fill:pink")
}

func main() {
    msize := 5.0
    fsize := msize * 40
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:white")
    female(width, height-50, fsize)
    male(100, 200, msize)
    canvas.End()
}
```

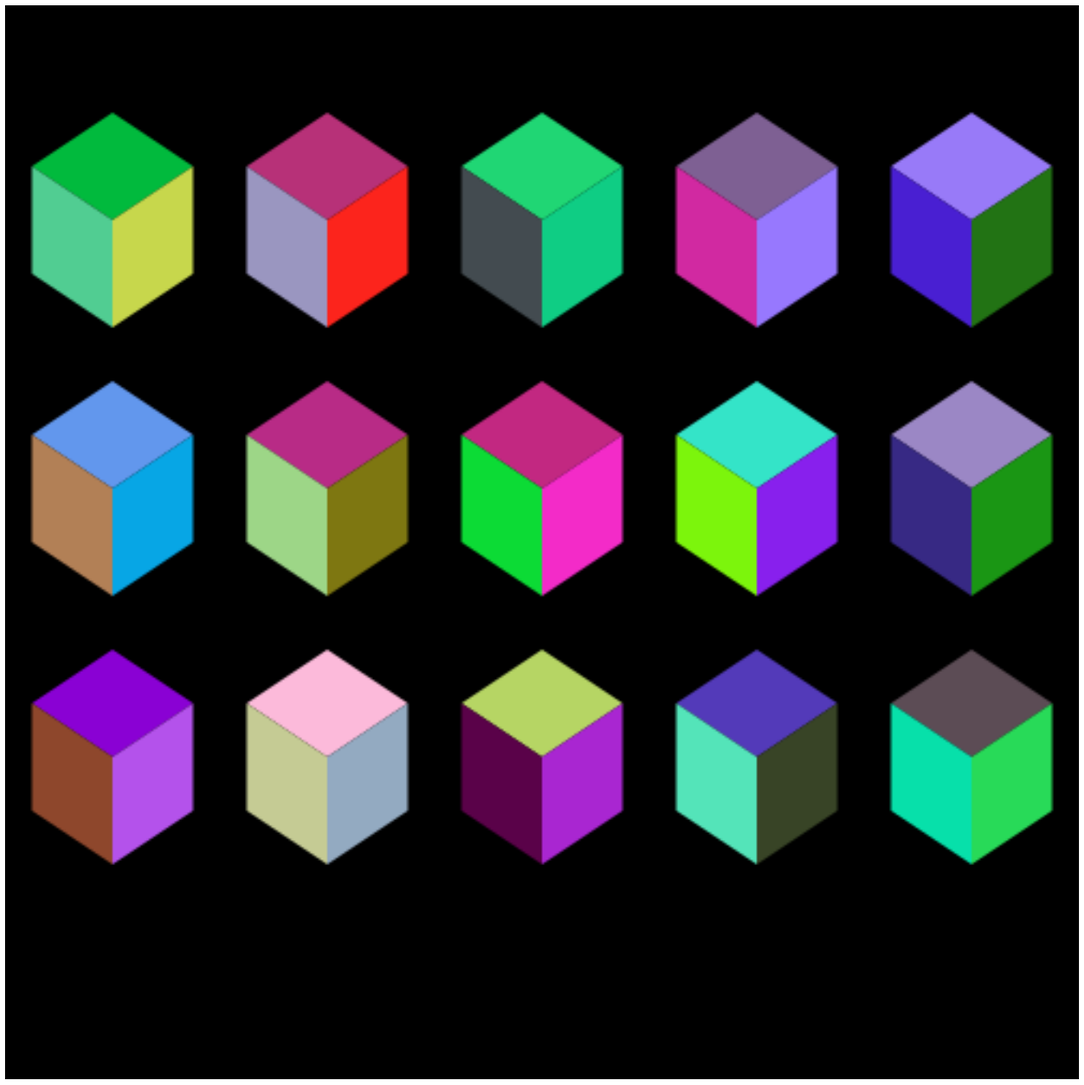
```
package main

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

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func male(x, y, w float64) {
    canvas.Ellipse(x, y, w, w/2, "fill:blue")
    canvas.Bezier(
        x-(w*8), y,
        x-(w*4), y-(w*4),
        x-(w*4), y+w,
        x-w, y, "stroke:blue;fill:none")
}

func female(x, y, w float64) {
    canvas.Circle(x, y, w, "fill:pink")
}

func main() {
    rand.Seed(time.Now().Unix())
    msize := 5.0
    fsize := msize * 40
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:white")
    female(width, height-50, fsize)
    for i := 0; i < 100; i++ {
        canvas.TranslateRotate((rand.Float64()*300)+100, (rand.Float64()*200)+200, rand.Float64()*45)
        male(0, 0, msize)
        canvas.Gend()
    }
    canvas.End()
}
```



```
package main

import (
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    canvas.Start(width, height)
    rand.Seed(time.Now().Unix())
    for i := 0; i < 100; i++ {
        fill := canvas.RGBA(
            rand.Intn(255),
            rand.Intn(255),
            rand.Intn(255),
            rand.Float64())
        canvas.Rect(
            rand.Float64()*width,
            rand.Float64()*height,
            rand.Float64()*100,
            rand.Float64()*100,
            fill)
    }
    canvas.End()
}
```

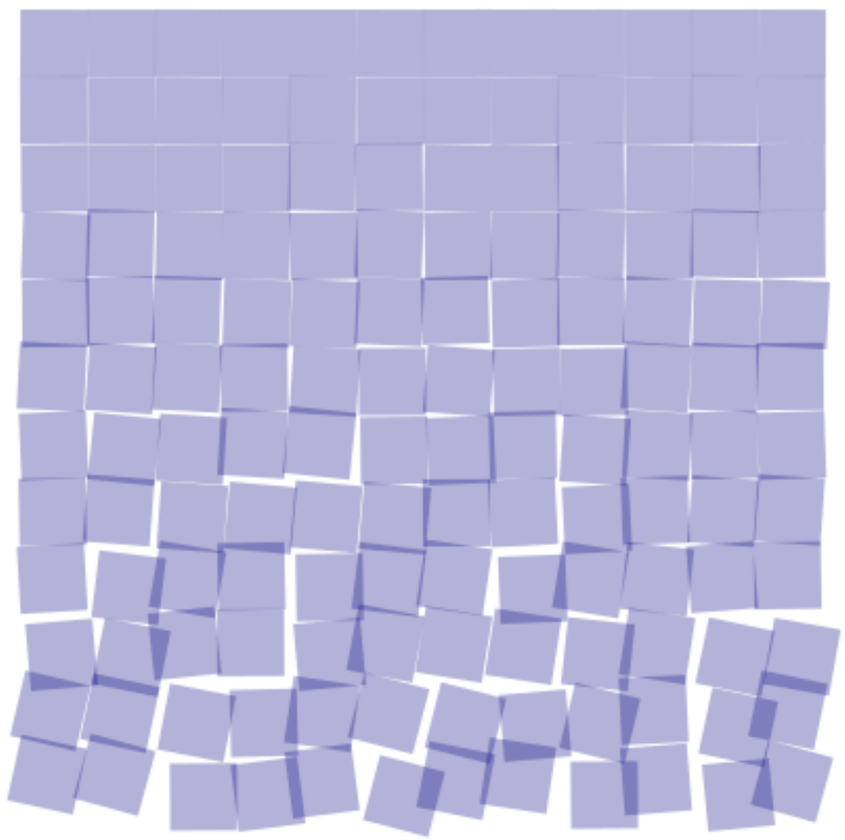
```
package main

import (
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    canvas.Start(width, height)
    rand.Seed(time.Now().Unix())
    for i := 0; i < 100; i++ {
        fill := canvas.RGBA(
            rand.Intn(255),
            rand.Intn(255),
            rand.Intn(255),
            rand.Float64())
        canvas.Ellipse(
            rand.Float64()*width,
            rand.Float64()*height,
            rand.Float64()*100,
            rand.Float64()*100,
            fill)
    }
    canvas.End()
}
```

```
package main

import (
    "math/rand"
    "os"
    "time"
    "github.com/ajstarks/gensvg"
)
var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)
func randcolor() string {
    return canvas.RGB(rand.Intn(255),rand.Intn(255),rand.Intn(255))
}
func rcube(x, y, l float64) {
    l2, l3, l4, l6, l8 := l*2, l*3, l*4, l*6, l*8
    tx := []float64{x, x + (l3), x, x - (l3), x}
    ty := []float64{y, y + (l2), y + (l4), y + (l2), y}
    lx := []float64{x - (l3), x, x, x - (l3), x - (l3)}
    ly := []float64{y + (l2), y + (l4), y + (l8), y + (l6), y + (l2)}
    rx := []float64{x + (l3), x + (l3), x, x, x + (l3)}
    ry := []float64{y + (l2), y + (l6), y + (l8), y + (l4), y + (l2)}
    canvas.Polygon(tx, ty, randcolor())
    canvas.Polygon(lx, ly, randcolor())
    canvas.Polygon(rx, ry, randcolor())
}
func main() {
    rand.Seed(time.Now().Unix())
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    xp, y := width/10, height/10
    n, hspace, vspace, size := 3, width/5, height/4, width/40
    for r := 0; r < n; r++ {
        for x := xp; x < width; x += hspace {
            rcube(x, y, size)
        }
        y += vspace
    }
    canvas.End()
}
```

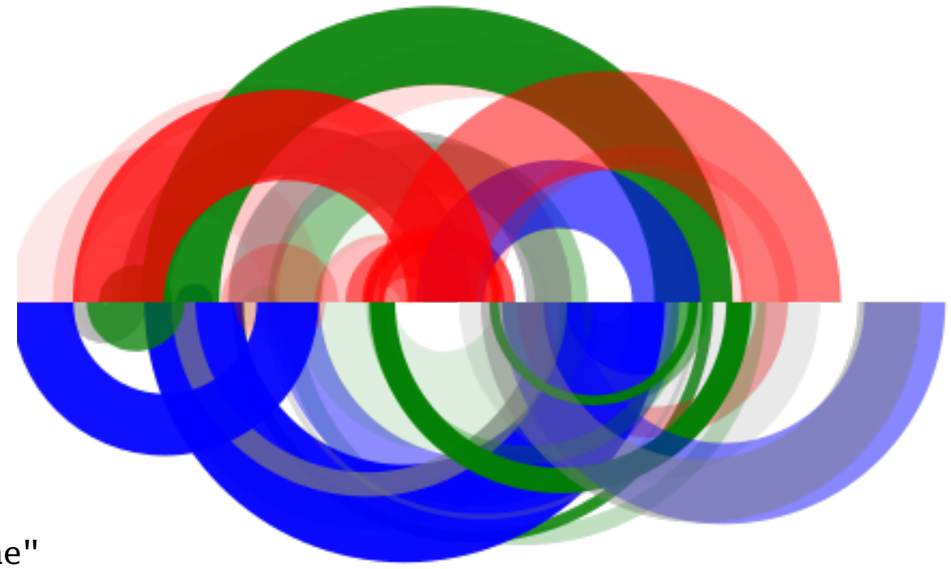


cube.go [23]

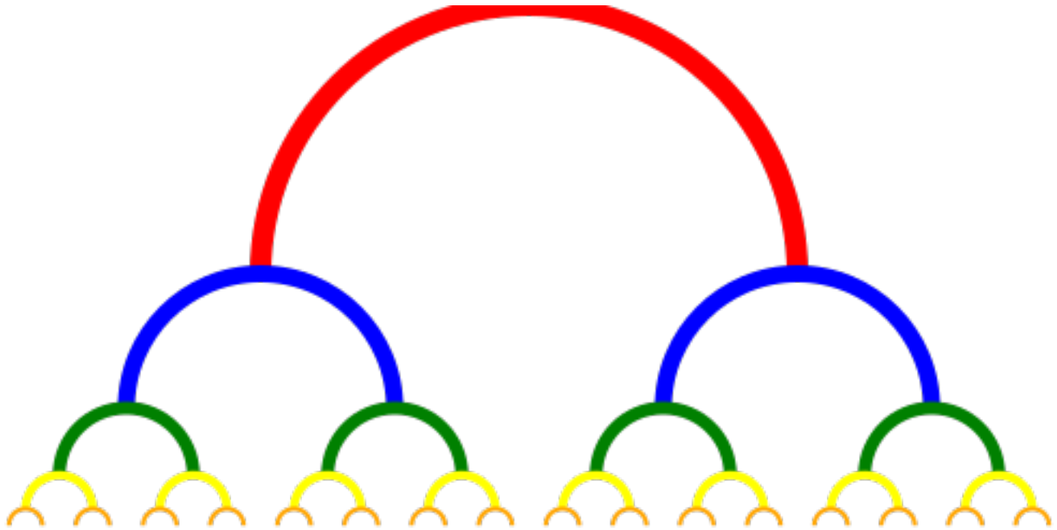
```
package main

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

func tloc(x, y, s float64, r, d float64) (float64, float64) {
    padding := 2 * s
    return (padding + (x * s) - (.5 * s) + (r * d)),
        (padding + (y * s) - (.5 * s) + (r * d))
}

func random(n float64) float64 {
    x := rand.Float64()
    if x < 0.5 {
        return -n * x
    }
    return n * x
}

func main() {
    columns, rows, sqrsize := 12.0, 12.0, 32.0
    rndStep, dampen := .22, 0.45
    width, height := (columns+4)*sqrsize, (rows+4)*sqrsize
    canvas := gensvg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:white")
    canvas.Gstyle("fill:rgb(0,0,127);fill-opacity:0.3")
    for y, randsum := 1.0, 0.0; y <= rows; y++ {
        randsum += float64(y) * rndStep
        for x := 1.0; x <= columns; x++ {
            tx, ty := tloc(x, y, sqrsize, random(randsum), dampen)
            canvas.TranslateRotate(tx, ty, random(randsum))
            canvas.CenterRect(0, 0, sqrsize, sqrsize)
            canvas.Gend()
        }
    }
    canvas.Gend()
    canvas.End()
}
```



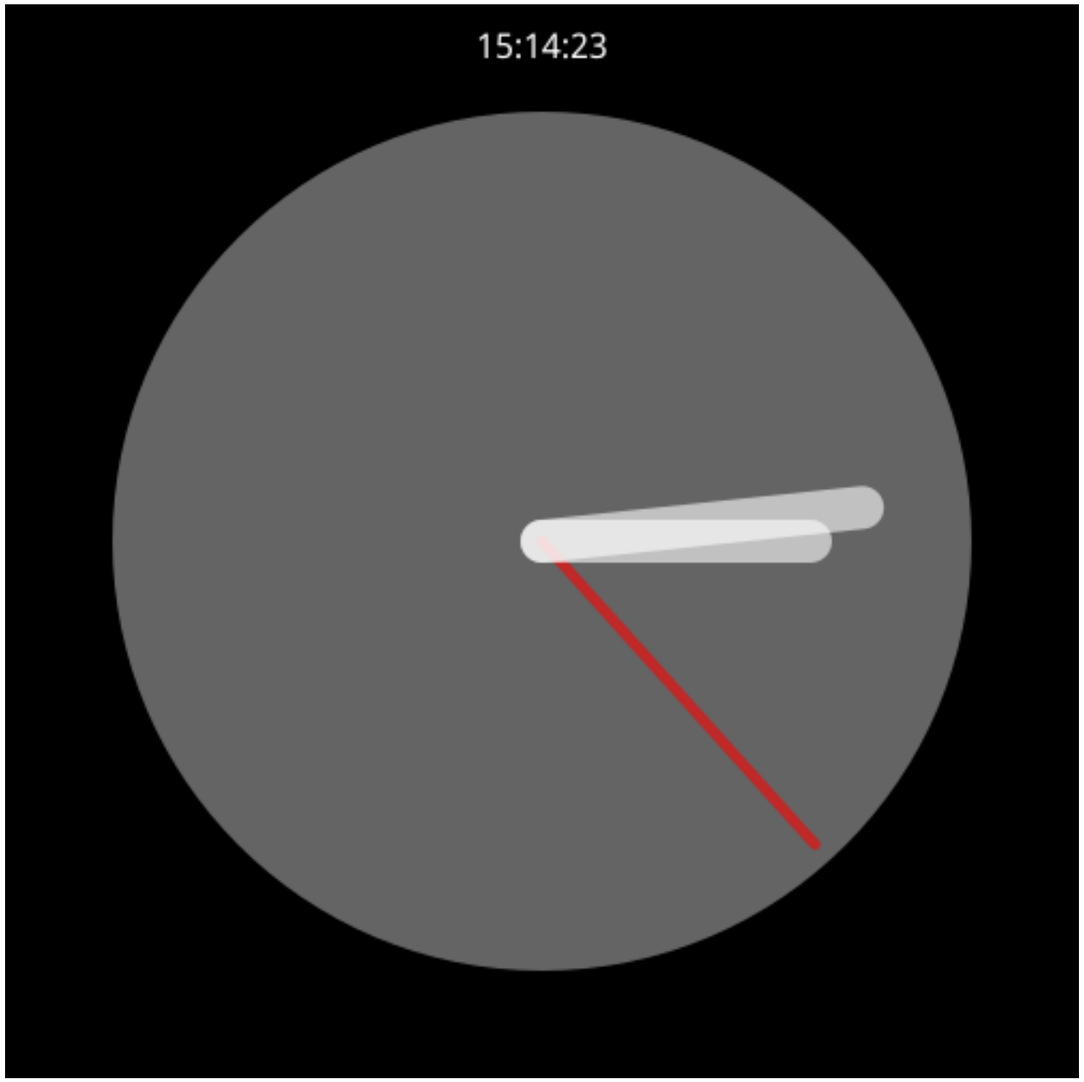
```
package main

import (
    "fmt"
    "math/rand"
    "os"
    "time"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func randarc(aw, ah, sw float64, f1, f2 bool) {
    colors := []string{"red", "green", "blue", "gray"}
    afmt := "stroke:%s;stroke-opacity:%.2f;stroke-width:%.2fpx;fill:none"
    begin, arclength := rand.Float64()*(aw), rand.Float64()*(aw)
    end := begin + arclength
    baseline := ah / 2
    al, cl := arclength/2, len(colors)
    canvas.Arc(begin, baseline, al, al, 0, f1, f2, end, baseline,
        fmt.Sprintf(afmt, colors[rand.Intn(cl)], rand.Float64(), rand.Float64()*(sw)))
}

func main() {
    rand.Seed(time.Now().Unix())
    canvas.Start(width, height)
    aw := width / 2
    maxstroke := height / 10
    for i := 0; i < 20; i++ {
        randarc(aw, height, maxstroke, false, true)
        randarc(aw, height, maxstroke, false, false)
    }
    canvas.End()
}
```



```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas   = gensvg.New(os.Stdout)
    width    = 500.0
    height   = 500.0
    maxlevel = 5
    colors   = []string{"red", "orange", "yellow", "green", "blue"}
)

func branch(x, y, r float64, level int) {
    astyle := fmt.Sprintf("fill:none;stroke:%s;stroke-width:%dpx", colors[level%maxlevel], level*2)
    canvas.Arc(x-r, y, r, r, 0, true, true, x+r, y, astyle)
    if level > 0 {
        branch(x-r, y+r/2, r/2, level-1)
        branch(x+r, y+r/2, r/2, level-1)
    }
}

// Example from "Generative Design", pg 414
func main() {
    canvas.Start(width, height)
    branch(0, 0, width/2, 6)
    canvas.End()
}
```

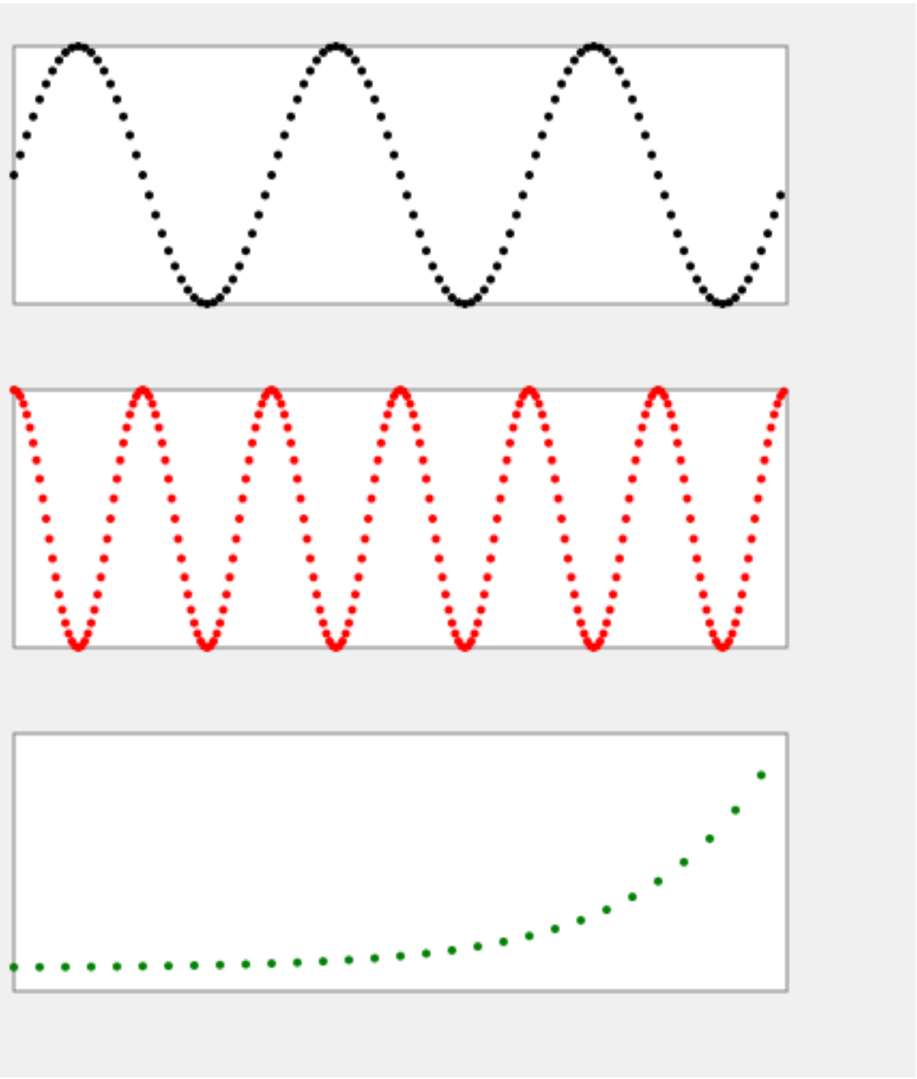


```
package main

import (
    "fmt"
    "github.com/ajstarks/gensvg"
    "math"
    "os"
    "time"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func vmap(value float64, l1 float64, h1 float64,
    l2 float64, h2 float64) float64 {
    return l2 + (h2-l2)*(value-l1)/(h1-l1)
}

// See: Processing (Reas and Fry), pg. 247
func main() {
    w2, h2 := width/2, height/2
    h, m, s := time.Now().Clock()
    sec := vmap(float64(s), 0, 60, 0, math.Pi*2) - math.Pi/2
    min := vmap(float64(m), 0, 60, 0, math.Pi*2) - math.Pi/2
    hour := vmap(float64(h%12), 0, 12, 0, math.Pi*2) - math.Pi/2
    secpct := (width) * 0.38
    minpct := (width) * 0.30
    hourpct := (width) * 0.25
    facepct := (width * 40) / 100
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Text(w2, 25, fmt.Sprintf("%02d:%02d:%02d", h, m, s), "text-anchor:middle;font-size:12pt;fill:white")
    canvas.Circle(w2, h2, facepct, canvas.RGB(100, 100, 100))
    canvas.Gstyle("stroke:white;stroke-width:20;stroke-opacity:0.6;stroke-linecap:round")
    canvas.Line(w2, h2, (math.Cos(sec)*secpct)+w2, (math.Sin(sec)*secpct)+h2, "stroke:red;stroke-width:5")
    canvas.Line(w2, h2, (math.Cos(min)*minpct)+w2, (math.Sin(min)*minpct)+h2)
    canvas.Line(w2, h2, (math.Cos(hour)*hourpct)+w2, (math.Sin(hour)*hourpct)+h2)
    canvas.Gend()
    canvas.End()
}
```



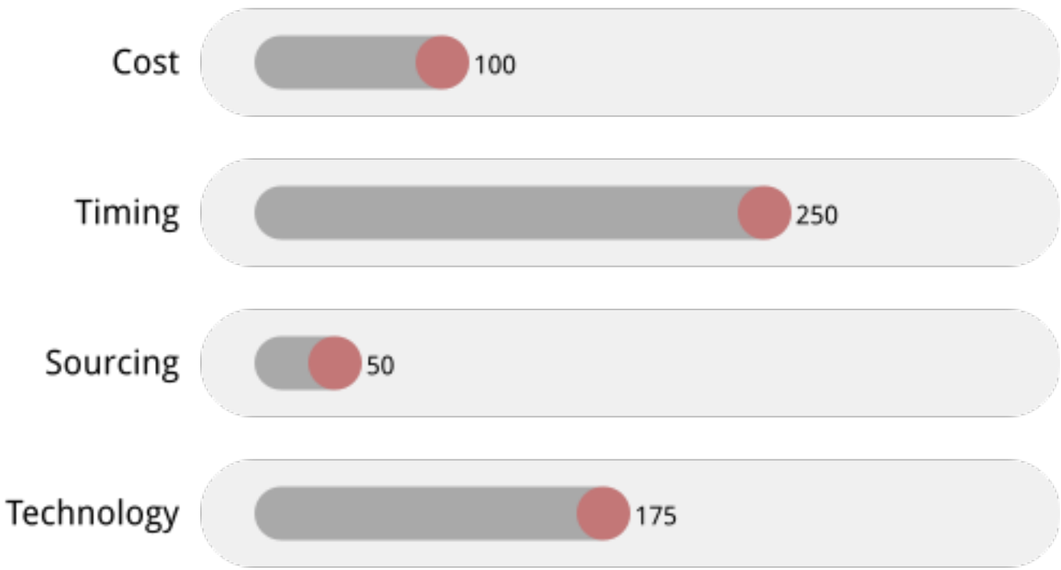
clock.go [27]

```
package main

import (
    "math"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func vmap(value float64, l1 float64, h1 float64,
    l2 float64, h2 float64) float64 {
    return l2 + (h2-l2)*(value-l1)/(h1-l1)
}

func plotfunc(left, top, w, h float64, min, max, fmin, fmax,
    interval float64, f func(float64) float64, style ...string) {
    canvas.Translate(0, top)
    canvas.Rect(left, 0, w, h, "fill:white;stroke:gray")
    for x := min; x < max; x += interval {
        dx := vmap(x, min, max, float64(left), float64(w+left))
        dy := vmap(f(x), fmin, fmax, 0, float64(h))
        canvas.Translate(0, (h - height))
        canvas.Circle(dx, height-dy, 2, style...)
        canvas.Gend()
    }
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:rgb(240,240,240)")
    plotfunc(80, 20, 360, 120, 0, 6*math.Pi, -1, 1, math.Pi/20, math.Sin)
    plotfunc(80, 180, 360, 120, 0, 12*math.Pi, -1, 1, math.Pi/20, math.Cos, "fill:red")
    plotfunc(80, 340, 360, 120, -3, 3, -2, 20, 0.2, math.Exp, "fill:green")
    canvas.End()
}
```

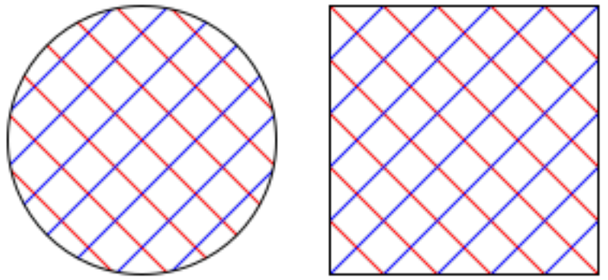


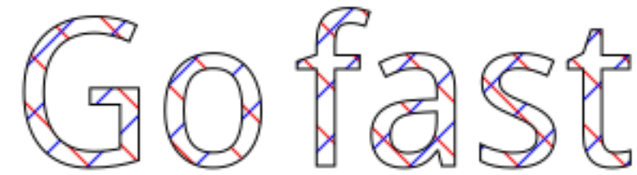
plotfunc.go [28]

```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

type Measure struct {
    name  string
    value float64
}

func (data *Measure) meter(x, y, w, h float64) {
    corner := h / 2
    inset := corner / 2
    canvas.Text(x-10, y+h/2, data.name, "text-anchor:end;baseline-shift:-33%")
    canvas.Roundrect(x, y, w, h, corner, corner, "fill:rgb(240,240,240)")
    canvas.Roundrect(x+corner, y+inset, data.value, h-(inset*2), inset, inset, "fill:darkgray")
    canvas.Circle(x+inset+data.value, y+corner, inset, "fill:red;fill-opacity:0.3")
    canvas.Text(x+inset+data.value+inset+2, y+h/2, fmt.Sprintf("%-0.f", data.value),
        "font-size:75%;text-anchor:start;baseline-shift:-33%")
}

func main() {
    items := []Measure{{"Cost", 100}, {"Timing", 250}, {"Sourcing", 50}, {"Technology", 175}}
    x, y, gutter, mh := 100.0, 50.0, 20.0, 50.0
    canvas.Start(width, height)
    canvas.Gstyle("font-family:sans-serif;font-size:12pt")
    for _, data := range items {
        data.meter(x, y, width-100, mh)
        y += mh + gutter
    }
    canvas.Gend()
    canvas.End()
}
```

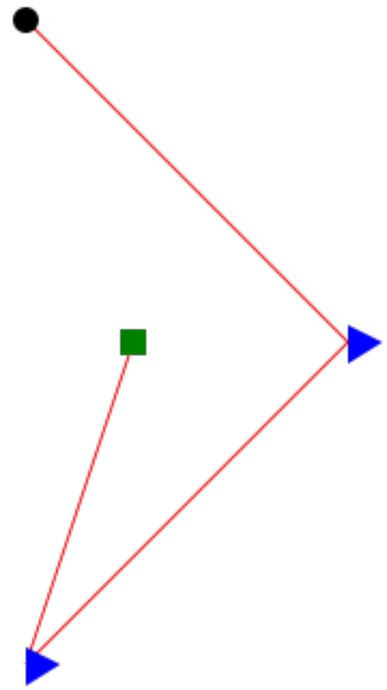


```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    pct := 5.0
    pw, ph := (width*pct)/100, (height*pct)/100
    canvas.Start(width, height)

    canvas.Def()
    canvas.Pattern("hatch", 0, 0, pw, ph, "user")
    canvas.Gstyle("fill:none;stroke-width:1")
    canvas.Path(fmt.Sprintf("M0,0 l%0.f,%0.f", pw, ph), "stroke:red")
    canvas.Path(fmt.Sprintf("M%0.f,0 l-%0.f,%0.f", pw, pw, ph), "stroke:blue")
    canvas.Gend()
    canvas.PatternEnd()
    canvas.DefEnd()

    x1 := width / 2
    x2 := (width * 4) / 5
    canvas.Gstyle("stroke:black; font-size: 72pt; text-anchor:middle; fill:url(#hatch)")
    canvas.Circle(x1, height/2, height/8)
    canvas.CenterRect(x2, height/2, height/4, height/4)
    canvas.Text(x1, height-50, "Go")
    canvas.Text(x2, height-50, "fast")
    canvas.Gend()
    canvas.End()
}
```

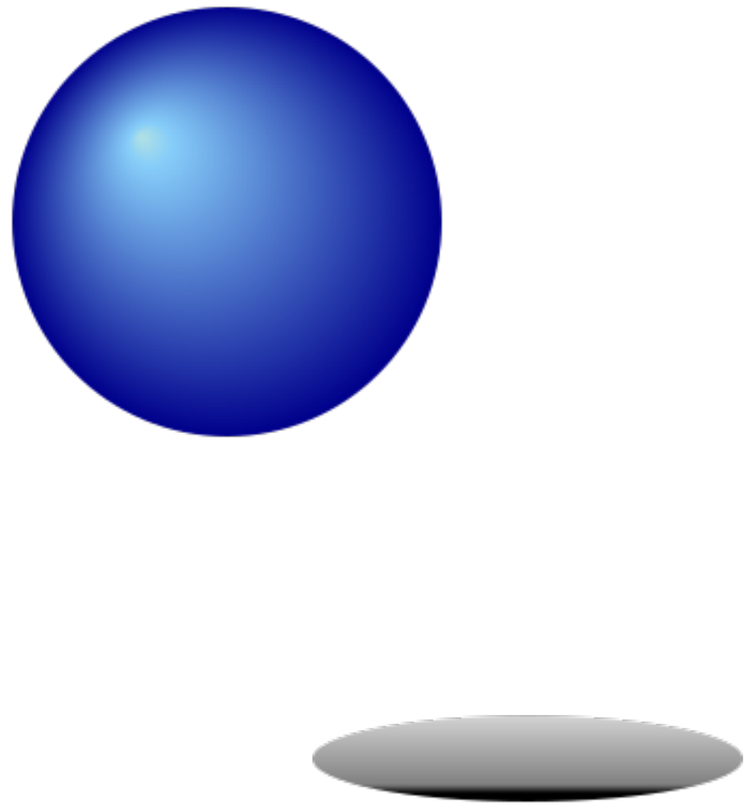




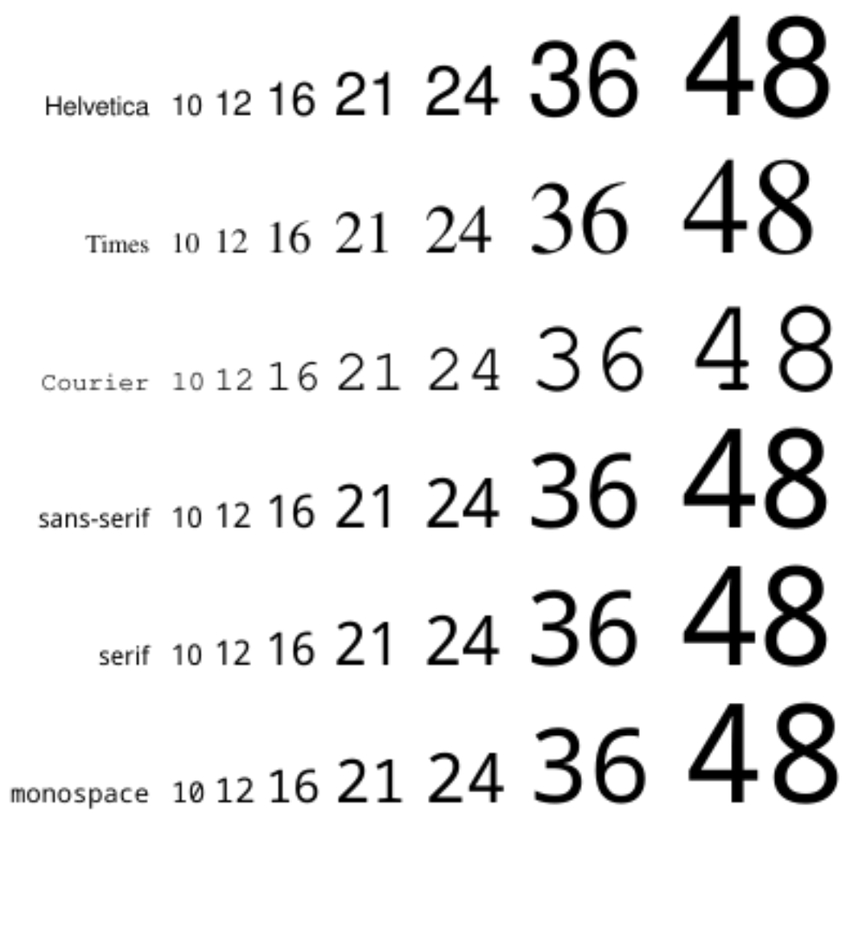
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {

    canvas.Start(width, height)
    canvas.Def()
    canvas.Marker("dot", 10, 10, 16, 16)
    canvas.Circle(10, 10, 6, "fill:black")
    canvas.MarkerEnd()

    canvas.Marker("box", 10, 10, 16, 16)
    canvas.CenterRect(10, 10, 12, 12, "fill:green")
    canvas.MarkerEnd()

    canvas.Marker("arrow", 4, 12, 26, 26)
    canvas.Path("M4,4 L4,22 L20,12 L4,4", "fill:blue")
    canvas.MarkerEnd()
    canvas.DefEnd()

    x := []float64{100, 250, 100, 150}
    y := []float64{100, 250, 400, 250}
    canvas.Polyline(x, y,
        `fill="none"`,
        `stroke="red"`,
        `marker-start="url(#dot)"`,
        `marker-mid="url(#arrow)"`,
        `marker-end="url(#box)"`)
    canvas.End()
}
```



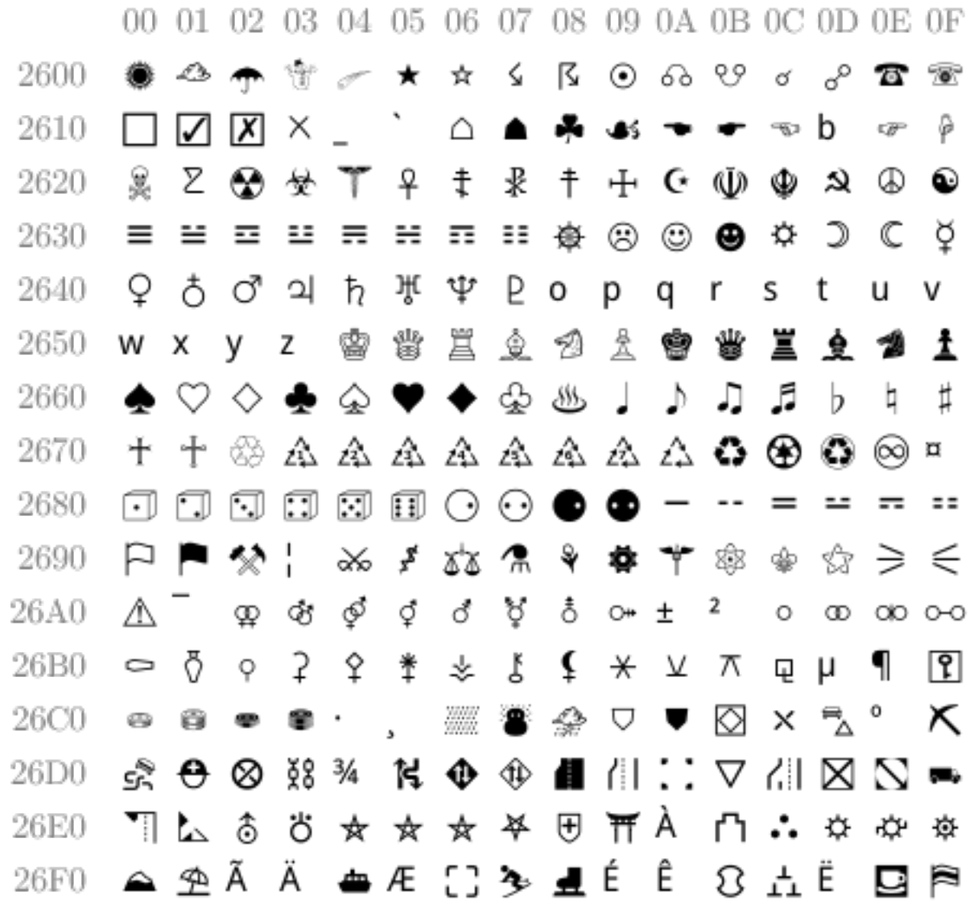
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    rg := []gensvg.Offcolor{
        {1, "powderblue", 1},
        {10, "lightskyblue", 1},
        {100, "darkblue", 1},
    }
    lg := []gensvg.Offcolor{
        {10, "black", 1},
        {20, "gray", 1},
        {100, "lightgray", 1},
    }
    canvas.Start(width, height)
    canvas.Def()
    canvas.RadialGradient("rg", 50, 50, 50, 30, 30, rg)
    canvas.LinearGradient("lg", 0, 100, 0, 0, lg)
    canvas.DefEnd()
    canvas.Circle(width/2, height-300, 100, "fill:url(#rg)")
    canvas.Ellipse(width-110, height-50, 100, 20, "fill:url(#lg)")
    canvas.End()
}
```



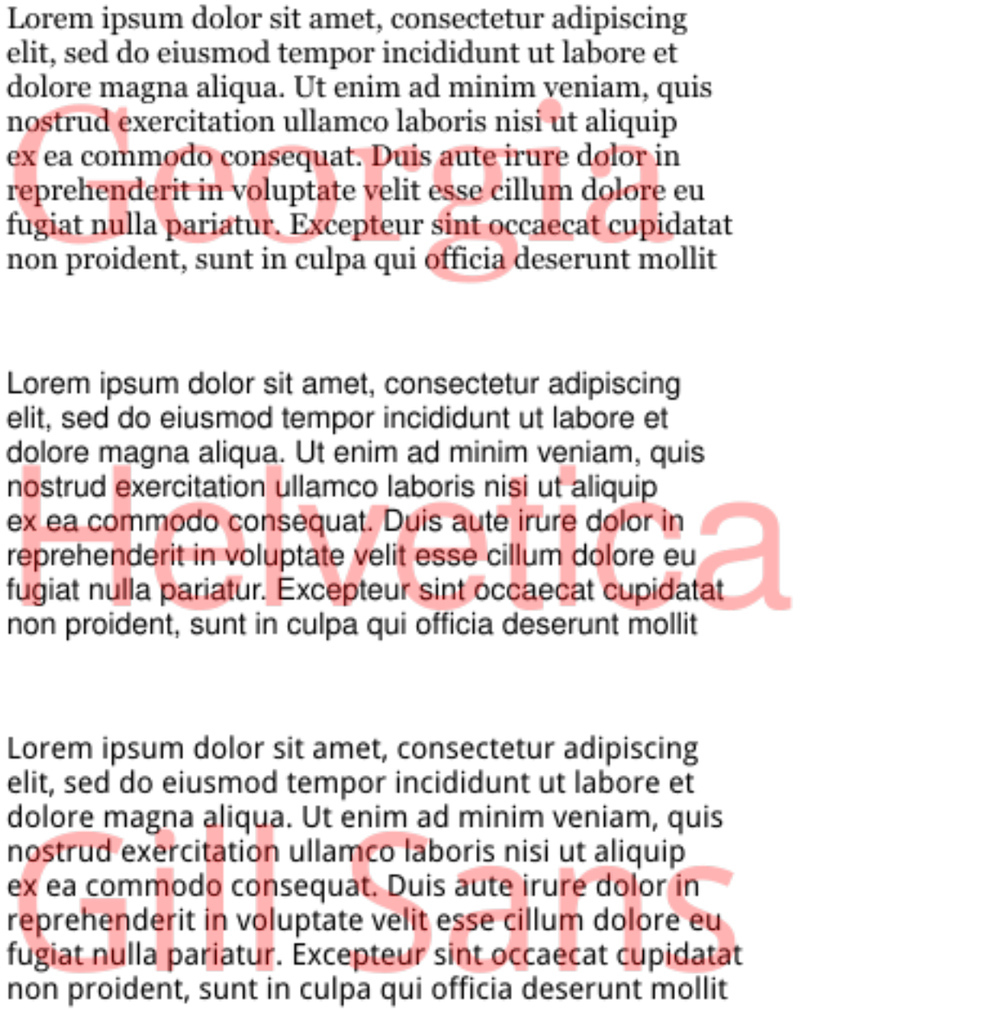
```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    fonts := []string{
        "Helvetica", "Times", "Courier",
        "sans-serif", "serif", "monospace",
    }
    sizes := []float64{10, 12, 16, 21, 24, 36, 48}
    largest := sizes[len(sizes)-1]
    gutter := largest + (largest / 3)
    margin := gutter * 2
    y := 100.0

    canvas.Start(width, height)
    for _, f := range fonts {
        x := margin
        canvas.Gstyle("font-family:" + f)
        canvas.Text(x-10, y, f, "text-anchor:end")
        for _, s := range sizes {
        canvas.Text(x, y, fmt.Sprintf("%0.f", s), fmt.Sprintf("font-size:%0.fpt", s))
            x += s * 2
        }
        canvas.Gend()
        y += gutter
    }
    canvas.End()
}
```



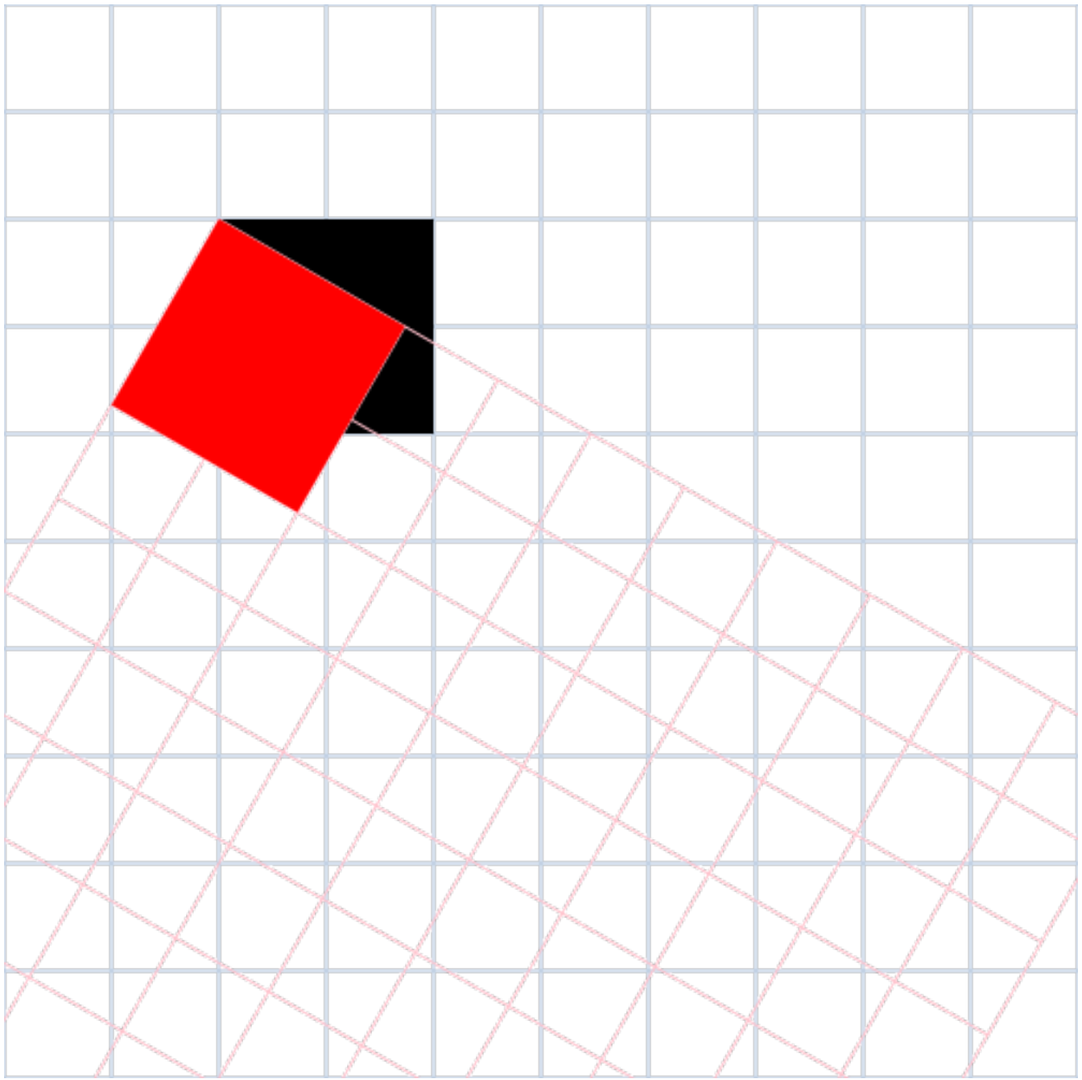
```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    top, left, fontsize := 50.0, 100.0, 16.0
    xoffset, yoffset := 25.0, 25.0
    rows, cols := 16, 16
    glyph := 0x2600
    font := "Symbola"
    stylefmt := "font-family:%s;font-size:%.2fpx;text-anchor:middle"
    canvas.Start(width, height)
    canvas.Gstyle(fmt.Sprintf(stylefmt, font, fontsize))

    x, y := left, top
    for r := 0; r < rows; r++ {
        canvas.Text(x-yoffset, y, fmt.Sprintf("%X", glyph), "text-anchor:end;fill:gray")
        for c := 0; c < cols; c++ {
            if r == 0 {
                canvas.Text(x, y-yoffset, fmt.Sprintf("%02X", c), "fill:gray")
            }
            canvas.Text(x, y, string(glyph))
            glyph++
            x += xoffset
        }
        x = left
        y += yoffset
    }
    canvas.Gend()
    canvas.End()
}
```



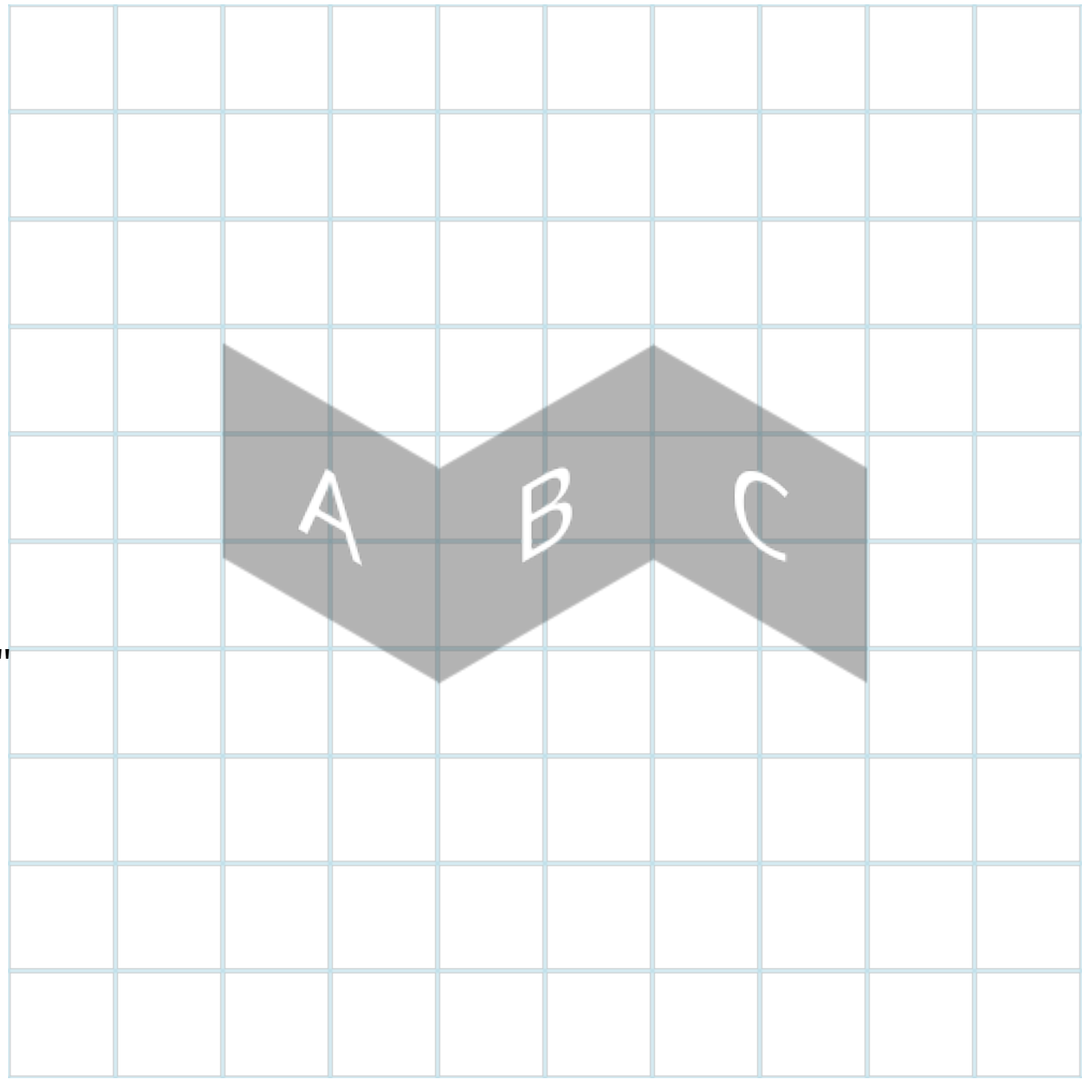
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    lorem := []string{
        "Lorem ipsum dolor sit amet, consectetur adipiscing",
        "elit, sed do eiusmod tempor incididunt ut labore et",
        "dolore magna aliqua. Ut enim ad minim veniam, quis",
        "nostrud exercitation ullamco laboris nisi ut aliquip",
        "ex ea commodo consequat. Duis aute irure dolor in",
        "reprehenderit in voluptate velit esse cillum dolore eu",
        "fugiat nulla pariatur. Excepteur sint occaecat cupidatat",
        "non proident, sunt in culpa qui officia deserunt mollit",
    }
    fontlist := []string{"Georgia", "Helvetica", "Gill Sans"}
    size, leading := 14.0, 16.0
    x, y := 50.0, 20.0
    tsize := float64(len(lorem))*leading + size*3
    canvas.Start(width, height)
    for _, f := range fontlist {
        canvas.Gstyle("font-family:" + f)
        canvas.Textlines(x, y, lorem, size, leading, "black", "start")
        canvas.Text(x, size+y+tsize/2, f, "fill-opacity:0.3;fill:red;font-size:750%")
        canvas.Gend()
        y += tsize
    }
    canvas.End()
}
```



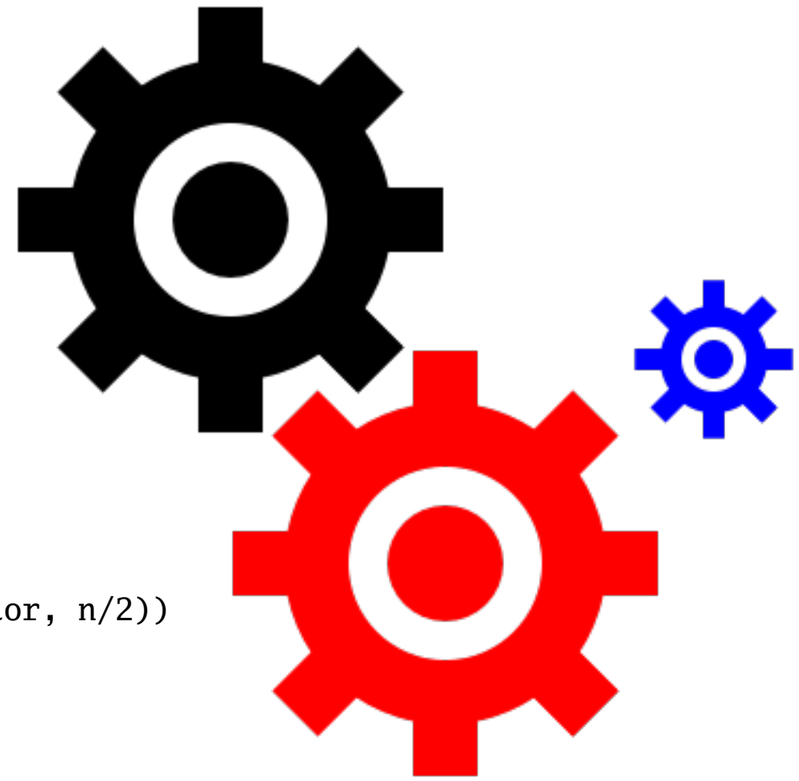
```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func tro() {
    canvas.Rect(100, 100, 100, 100)
    canvas.TranslateRotate(100, 100, 30)
    canvas.Grid(0, 0, width, height, 50, "stroke:pink")
    canvas.Rect(0, 0, 100, 100, "fill:red")
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    canvas.Grid(0, 0, width, height, 50, "stroke:lightsteelblue")
    tro()
    canvas.End()
}
```



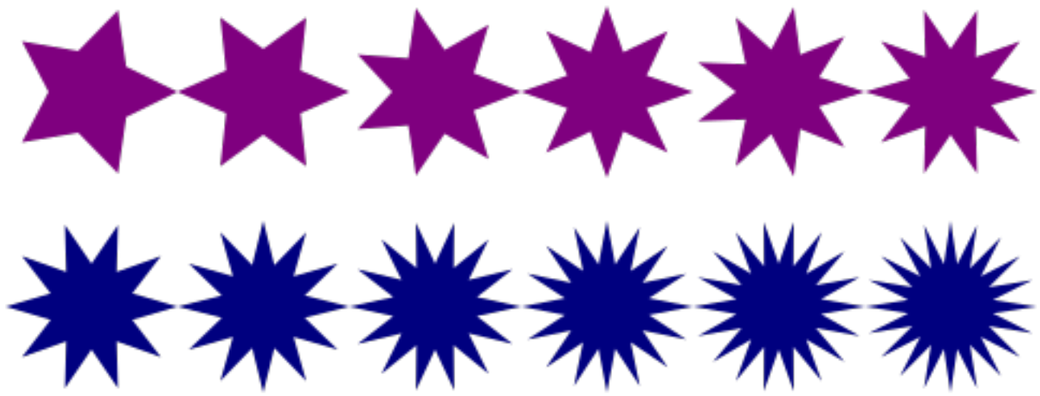
```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func sky(x, y, w, h, a float64, s string) {
    tfmt := "font-family:sans-serif;font-size:%.2fpx;text-anchor:middle"
    canvas.Gstyle(fmt.Sprintf(tfmt, w/2))
    canvas.SkewXY(0, float64(a))
    canvas.Rect(x, y, w, h, "fill:black;fill-opacity:0.3")
    canvas.Text(x+w/2, y+h/2, s, "fill:white;baseline-shift:-33%")
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    canvas.Grid(0, 0, width, height, 50, "stroke:lightblue")
    sky(100, 100, 100, 100, 30, "A")
    sky(200, 332, 100, 100, -30, "B")
    sky(300, -15, 100, 100, 30, "C")
    canvas.End()
}
```



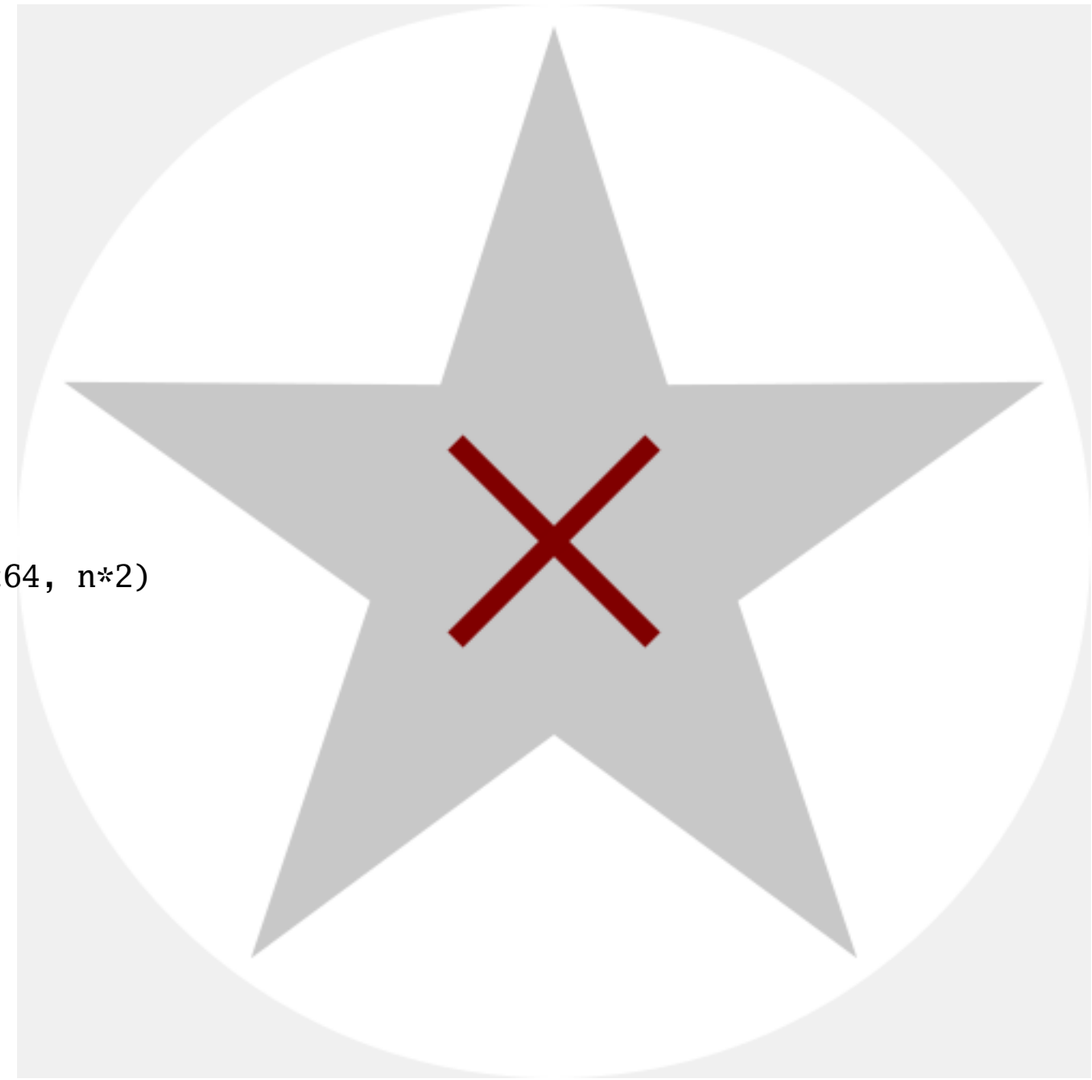
```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func gear(x, y, w, h, n, l, m float64, color string) {
    canvas.Gstyle(fmt.Sprintf("fill:none;stroke:%s;stroke-width:%.2f", color, n/2))
    canvas.Circle(x+w/2, y+h/2, n)
    canvas.Circle(x+w/2, y+h/2, n/5, "fill:"+color)
    ai := 360 / float64(m)
    for a := 0.0; a <= 360.0; a += ai {
        canvas.TranslateRotate(x+w/2, y+h/2, a)
        canvas.Line(n-l, n-l, n+l, n+l)
        canvas.Gend()
    }
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    gear(0, 0, 250, 250, 60, 10, 8, "black")
    gear(100, 160, 250, 250, 60, 10, 8, "red")
    gear(300, 140, 100, 100, 20, 6, 8, "blue")
    canvas.End()
}
```



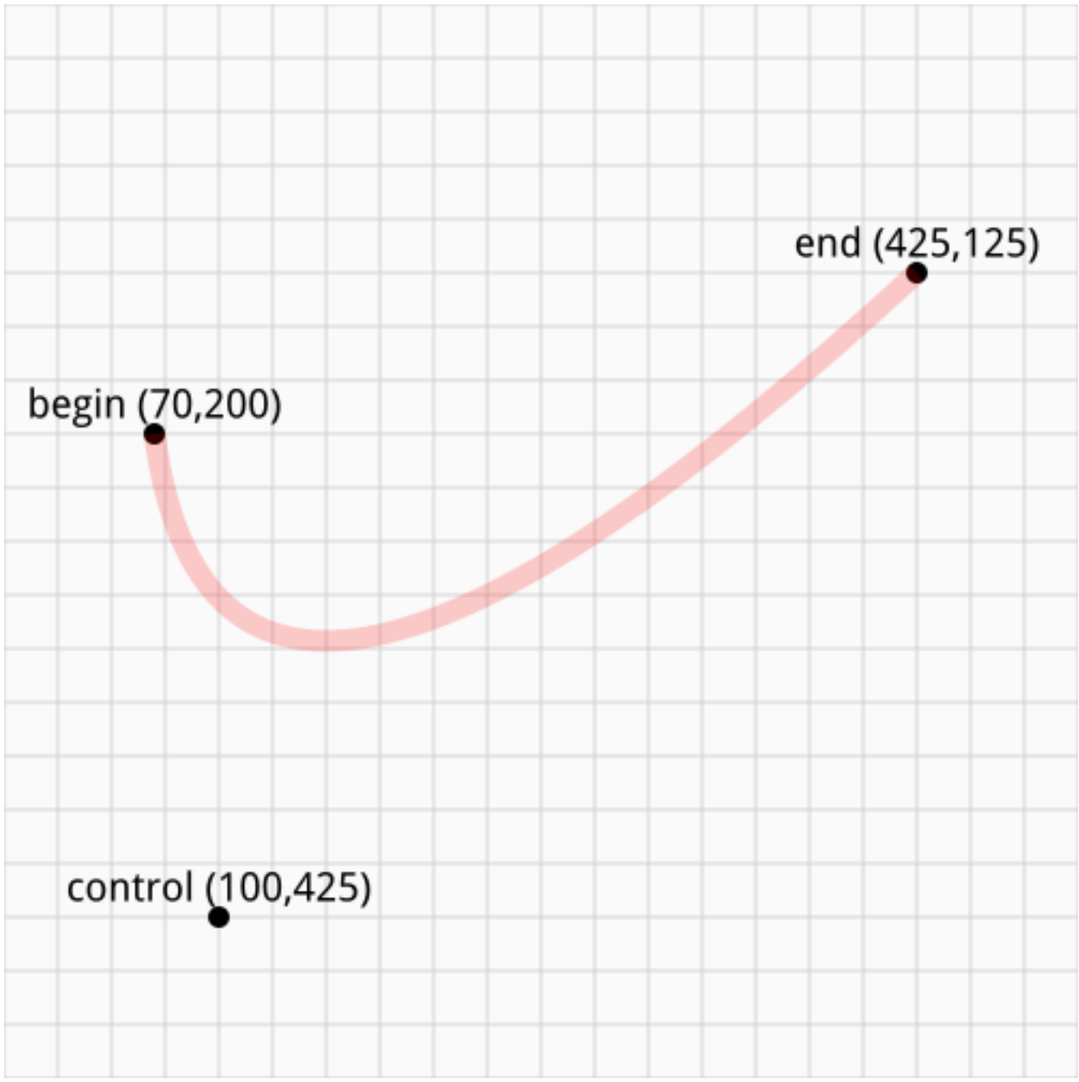
```
package main

import (
    "math"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

// See: http://vormplus.be/blog/article/processing-month-day-4-stars
func star(xp, yp float64, n int, inner, outer float64, style string) {
    xv, yv := make([]float64, n*2), make([]float64, n*2)
    angle := math.Pi / float64(n)
    for i := 0; i < n*2; i++ {
        fi := float64(i)
        if i%2 == 0 {
            xv[i] = math.Cos(angle*fi) * outer
            yv[i] = math.Sin(angle*fi) * outer
        } else {
            xv[i] = math.Cos(angle*fi) * inner
            yv[i] = math.Sin(angle*fi) * inner
        }
    }
    canvas.Translate(xp, yp)
    canvas.Polygon(xv, yv, style)
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    for x, i := 50.0, 5; i <= 10; i++ {
        star(x, 200, i, 20, 40, canvas.RGB(127, 0, 127))
        star(x, 300, i*2, 20, 40, canvas.RGB(0, 0, 127))
        x += 80
    }
    canvas.End()
}
```



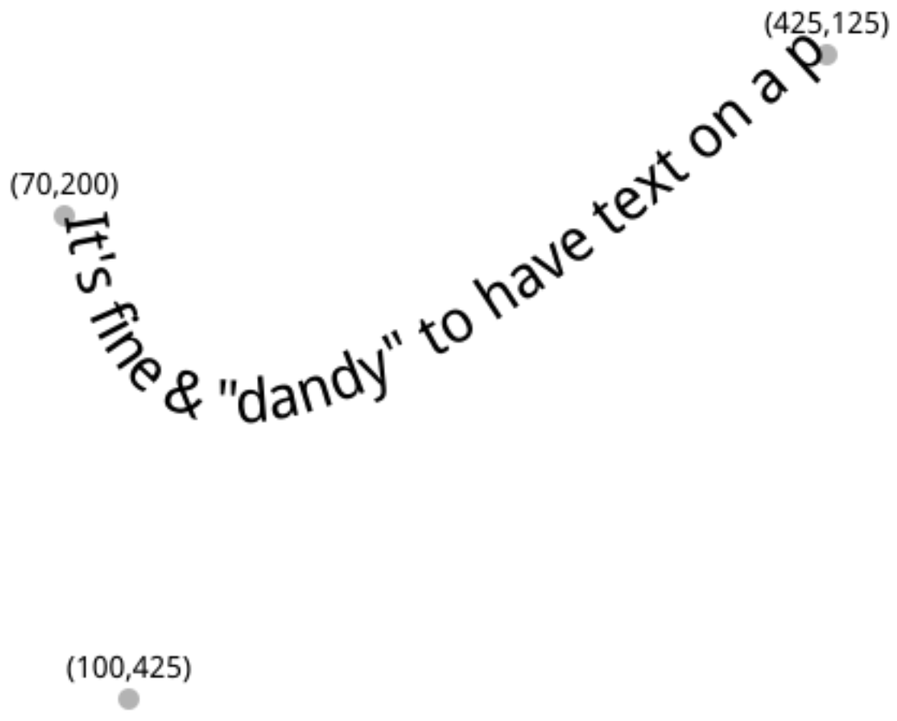
```
package main

import (
    "math"
    "os"
    "github.com/ajstarks/gensvg"
)
var canvas, width, height = gensvg.New(os.Stdout), 500.0, 500.0

func polar(cx, cy, r, t float64) (float64, float64) {
    return cx + (r * math.Cos(t)), cy + (r * math.Sin(t))
}

func star(x, y float64, n int, inner, outer float64, style string) {
    t, xv, yv := math.Pi/float64(n), make([]float64, n*2), make([]float64, n*2)
    for i := 0; i < n*2; i++ {
        if i%2 == 0 {
            xv[i], yv[i] = polar(0, 0, outer, t*float64(i))
        } else {
            xv[i], yv[i] = polar(0, 0, inner, t*float64(i))
        }
    }
    canvas.TranslateRotate(x, y, 54)
    canvas.Polygon(xv, yv, style)
    canvas.Gend()
}

func aline(x, y, r, a1, a2 float64) {
    x1, y1 := polar(x, y, r, a1)
    x2, y2 := polar(x, y, r, a2)
    canvas.Line(x1, y1, x2, y2, "stroke:maroon;stroke-width:10")
}

func main() {
    x, y, p4, r := width/2, height/2, math.Pi/4, 65.0
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, canvas.RGB(240, 240, 240))
    canvas.Circle(x, y, width/2, canvas.RGB(255, 255, 255))
    star(x, y, 5, 90, 240, canvas.RGB(200, 200, 200))
    aline(x, y, r, p4, 5*p4)
    aline(x, y, r, 3*p4, 7*p4)
    canvas.End()
}
```



```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func coord(x, y, size float64, label string) {
    tstyle := "text-anchor:middle;font-size:14pt"
    offset := size + (size / 2)
    canvas.Text(x, y-offset,
        fmt.Sprintf("%s (%d,%d)", label, int(x), int(y)), tstyle)
    canvas.Circle(x, y, size)
}

func showcurve(bx, by, cx, cy, ex, ey float64) {
    dotsize := 5.0
    sw := dotsize * 2
    cfmt := "stroke:%s;stroke-width:%.2f;fill:none;stroke-opacity:%.2f"
    style := fmt.Sprintf(cfmt, "red", sw, 0.2)
    coord(bx, by, dotsize, "begin")
    coord(ex, ey, dotsize, "end")
    coord(cx, cy, dotsize, "control")
    canvas.Qbez(bx, by, cx, cy, ex, ey, style)
}

func main() {
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:rgb(250,250,250)")
    canvas.Grid(0, 0, width, height, 25, "stroke:lightgray")
    showcurve(70, 200, 100, 425, 425, 125)
    canvas.End()
}
```



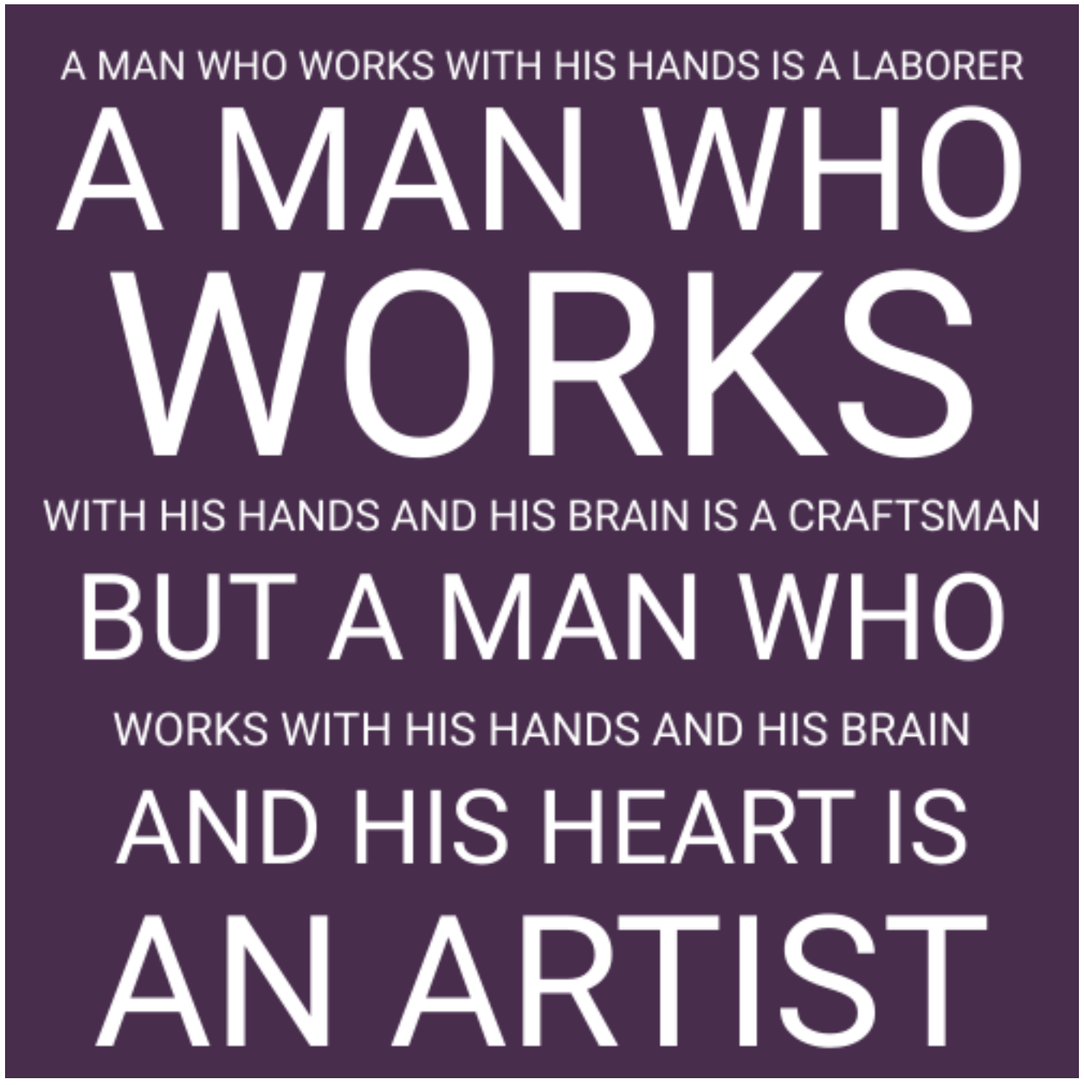
```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func coord(x, y, size float64) {
    offset := size * 2
    canvas.Text(x, y-offset, fmt.Sprintf("(%d,%d)", int(x), int(y)),
        "font-size:50%;text-anchor:middle")
    canvas.Circle(x, y, size, "fill-opacity:0.3")
}

func makepath(x, y, sx, sy, cx, cy, ex, ey float64, id, text string) {
    canvas.Def()
    canvas.Qbez(sx, sy, cx, cy, ex, ey, `id="`+id+`"`)
    canvas.DefEnd()
    canvas.Translate(x, y)
    canvas.Textpath(text, "#"+id)
    coord(sx, sy, 5)
    coord(ex, ey, 5)
    coord(cx, cy, 5)
    canvas.Gend()
}

func main() {
    message := `It's fine & "dandy" to have text on a path`
    canvas.Start(width, height)
    canvas.Gstyle("font-family:serif;font-size:21pt")
    makepath(0, 0, 70, 200, 100, 425, 425, 125, "tpath", message)
    canvas.Gend()
    canvas.End()
}
```



```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func main() {
    canvas.Start(width, height)
    opacity := 1.0
    for x := 0.0; x < width; x += 100 {
        canvas.Image(x, 100, 100, 124, "gopher.jpg", fmt.Sprintf("opacity:%.2f", opacity))
        opacity -= 0.15
    }
    canvas.End()
}
```

```
package main

import (
    "fmt"
    "os"
    "github.com/ajstarks/gensvg"
)
var canvas = gensvg.New(os.Stdout)

func main() {
    gutter, nc, iw, ih := 10.0, 2.0, 200.0, 112.0
    pw, ph := (iw*nc)+gutter*(nc+1), (ih*3)+gutter*4
    canvas.Start(pw, ph)
    canvas.Def()
    canvas.Filter("f0")
    canvas.Saturate(1.0)
    canvas.Fend()
    canvas.Filter("f1")
    canvas.FeComponentTransfer()
    canvas.FeFuncTable("G", []float64{0, 0.5, 0.6, 0.85, 1.0})
    canvas.FeCompEnd()
    canvas.Fend()
    for i, b := 0, 0.0; b < 20.0; b += 2.0 {
        canvas.Filter(fmt.Sprintf("blur%d", i))
        canvas.Blur(b)
        canvas.Fend()
        i++
    }
    canvas.DefEnd()
    x, y := gutter, gutter
    canvas.Gstyle("text-anchor:middle;fill:white;font-family:sans-serif;font-size:24pt")
    for i, f := range []string{"f0", "f1", "blur1", "blur2"} {
        if i != 0 && i%int(nc) == 0 {
            x = gutter
            y += ih + gutter
        }
        canvas.Image(x, y, int(iw), int(ih), "maple.jpg", "filter:url(#"+f+")")
        canvas.Text(x+iw/2, y+ih/2, f)
        x += iw + gutter
    }
    canvas.Gend()
    canvas.End()
}
```

```
package main

import (
    "fmt"
    "os"

    "github.com/ajstarks/gensvg"
)

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

func tf(x, y float64, s string, size float64) {
    canvas.Text(x, y, s, fmt.Sprintf("font-size:%gpt", size))
}

func main() {
    x, y := width/2, 35.0
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, canvas.RGB(72, 45, 77))
    canvas.Gstyle("font-family:Roboto;fill:white;text-anchor:middle")
    tf(x, y, "A MAN WHO WORKS WITH HIS HANDS IS A LABORER", 14)
    y += 70
    tf(x, y, "A MAN WHO", 60)
    y += 105
    tf(x, y, "WORKS", 90)
    y += 35
    tf(x, y, "WITH HIS HANDS AND HIS BRAIN IS A CRAFTSMAN", 15)
    y += 60
    tf(x, y, "BUT A MAN WHO", 42)
    y += 40
    tf(x, y, "WORKS WITH HIS HANDS AND HIS BRAIN", 16)
    y += 55
    tf(x, y, "AND HIS HEART IS", 36)
    y += 85
    tf(x, y, "AN ARTIST", 64)
    canvas.Gend()
    canvas.End()
}
```



```
package main

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

var (
    canvas = gensvg.New(os.Stdout)
    width  = 500.0
    height = 500.0
)

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()
}
```

```
package main

import (
    "os"

    "github.com/ajstarks/gensvg"
)

func main() {
    width, height := 500.0, 500.0
    csize := width / 20
    duration := 5.0
    repeat := 10.0

    canvas := gensvg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Arc(0, 250, 10, 10, 0, false, true, 500, 250,
        `id="top"`, `fill="none"`, `stroke="red"`)
    canvas.Arc(0, 250, 10, 10, 0, true, false, 500, 250,
        `id="bot"`, `fill="none"`, `stroke="blue"`)
    canvas.Circle(0, 0, csize, `fill="red"`, `id="red-dot"`)
    canvas.Circle(0, 0, csize, `fill="blue"`, `id="blue-dot"`)
    canvas.AnimateMotion("#red-dot", "#top", duration, repeat)
    canvas.AnimateMotion("#blue-dot", "#bot", duration, repeat)
    canvas.End()
}
```