

Deck



a Go package for presentations

DECK: a package for presentations

Deck is a package written in Go

That uses a singular markup language

With elements for text, lists, code, and graphics

All layout and sizes are expressed as percentages

Clients are interactive or create formats like PDF or SVG

Elements

Hello, World

A block of text, word-wrapped to a specified width. You may specify size, font, color, and opacity.

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, World")
}
```

<text>...</text>

bullet

- Point A
- Point B
- Point C
- Point D

plain

First item

Second item

The third item

and the last thing

number

1. This

2. That

3. The other

4. One more

```
<list>...</list>
```

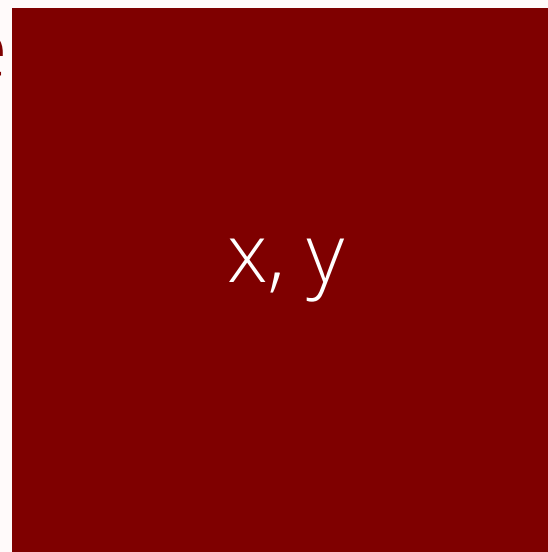
height



width

```
<image .../>
```

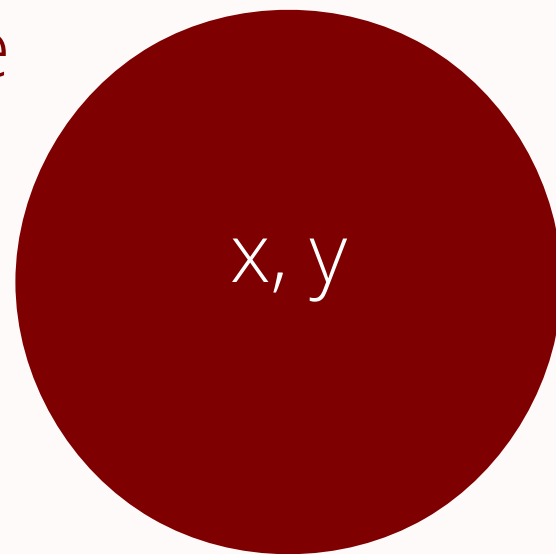
height (relative
to element
or canvas
width)



width

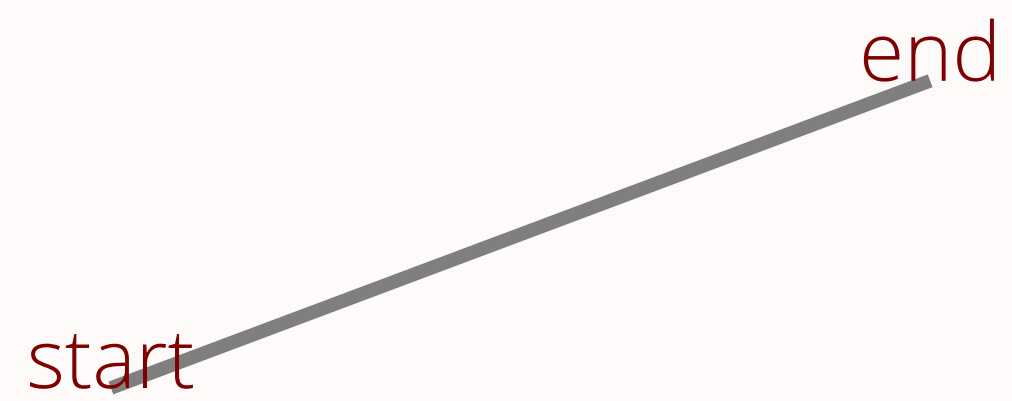
```
<rect ... />
```

height (relative
to element
or canvas
width)



width

```
<ellipse .../>
```

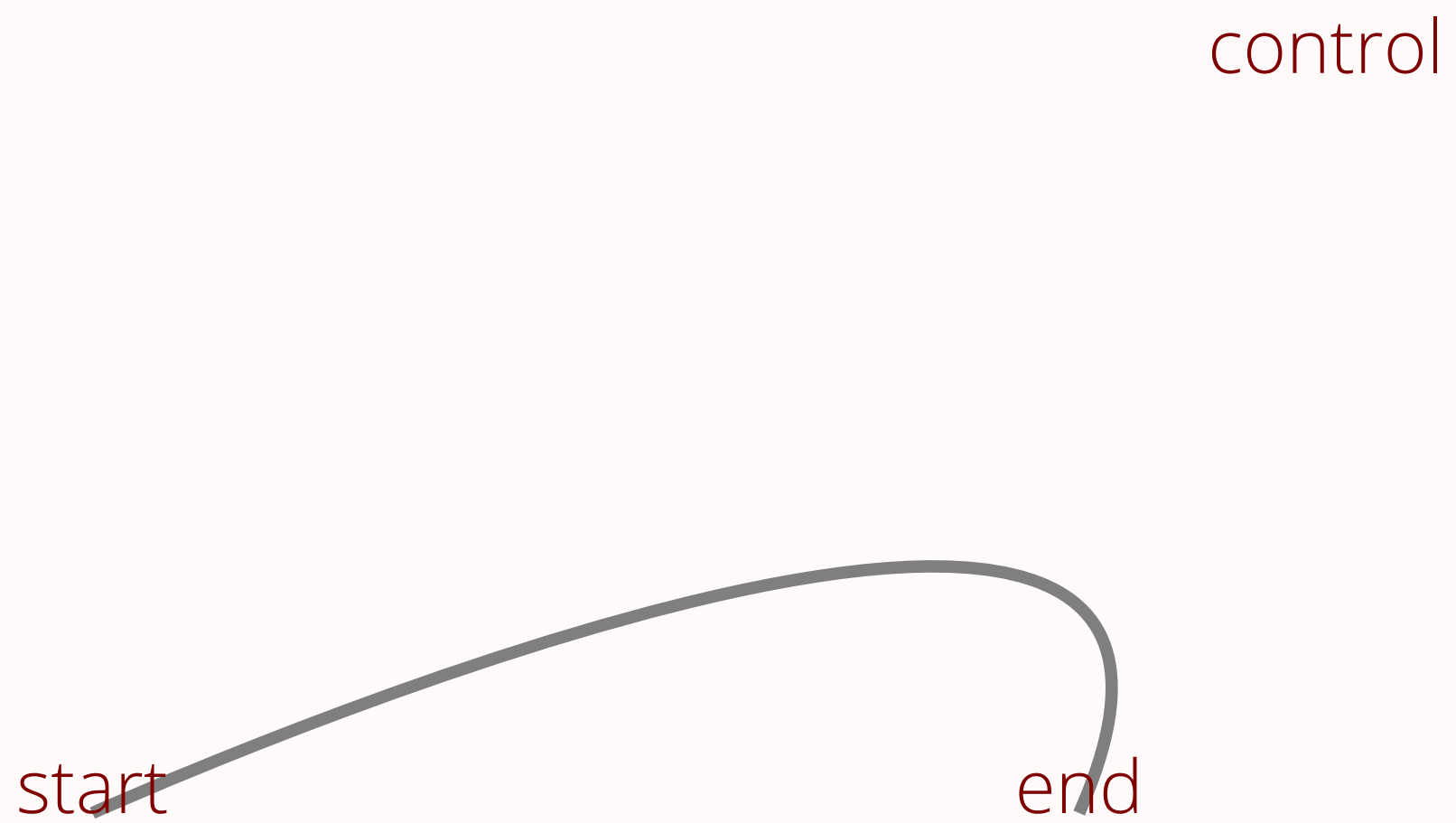
`<line .../>`

angle2 (90 deg)

x, y angle1 (0 deg)

A diagram showing a 90-degree arc. The arc starts at a point labeled 'x, y' and ends at a point labeled 'angle1 (0 deg)'. The arc is drawn in a counter-clockwise direction. The label 'angle2 (90 deg)' is placed at the end of the arc.

<arc ... />



<curve .../>

Markup and Layout

Start the deck
Set the canvas size
Begin a slide
Place an image
Draw some text
Make a bullet list

End the list
Draw a line
Draw a rectangle
Draw an ellipse
Draw an arc
Draw a quadratic bezier
End the slide
End of the deck

```
<deck>  
  <canvas width="1024" height="768" />  
  <slide bg="white" fg="black">  
    <image xp="70" yp="60" width="256" height="179" name="work.png" caption="Desk"/>  
    <text xp="20" yp="80" sp="3">Deck uses these elements</text>  
    <list xp="20" yp="70" sp="2" type="bullet">  
      <li>text, list, image</li>  
      <li>line, rect, ellipse</li>  
      <li>arc, curve</li>  
    </list>  
    <line xp1="20" yp1="10" xp2="30" yp2="10"/>  
    <rect xp="35" yp="10" wp="4" hr="75" color="rgb(127,0,0)"/>  
    <ellipse xp="45" yp="10" wp="4" hr="75" color="rgb(0,127,0)"/>  
    <arc xp="55" yp="10" wp="4" hp="3" a1="0" a2="180" color="rgb(0,0,127)"/>  
    <curve xp1="60" yp1="10" xp2="75" yp2="20" xp3="70" yp3="10" />  
  </slide>  
</deck>
```

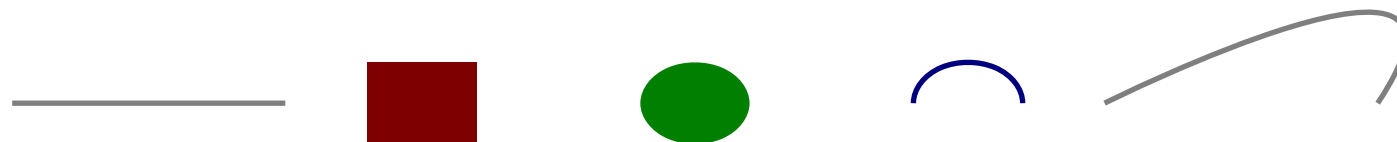
Anatomy of a Deck

Deck uses these elements

- text, list, image
- line, rect, ellipse
- arc, curve



Desk



Text and List Markup

Position, size <text xp="..." yp="..." sp="...">

Block of text <text ... type="block">

Lines of code <text ... type="code">

Attributes <text ... color="..." opacity="..." font="..." align="...">

Position, size <list xp="..." yp="..." sp="...">

Bullet list <list ... type="bullet">

Numbered list <list ... type="number">

Attributes <list ... color="..." opacity="..." font="..." align="...">

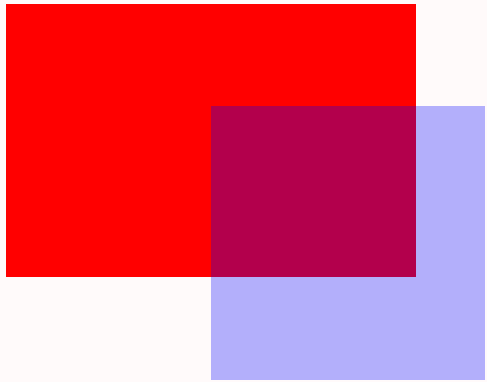
Common Attributes for text and list

xp	horizontal percentage
yp	vertical percentage
sp	font size percentage
type	"bullet", "number" (list), "block", "code" (text)
align	"left", "middle", "end"
color	SVG names ("maroon"), or RGB "rgb(127,0,0)"
opacity	percent opacity (0-100, transparent - opaque)
font	"sans", "serif", "mono"

Graphics Markup

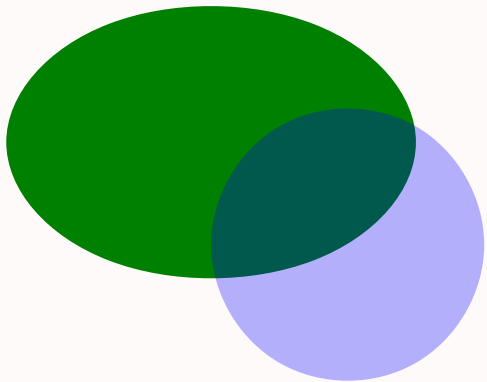


```
<line xp1="5" yp1="75" xp2="20" yp2="70" sp="0.2"/>
```



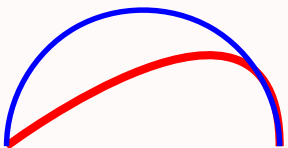
```
<rect xp="10" yp="60" wp="15" hr="66.6" color="red"/>
```

```
<rect xp="15" yp="55" wp="10" hr="100" color="blue" opacity="30"/>
```



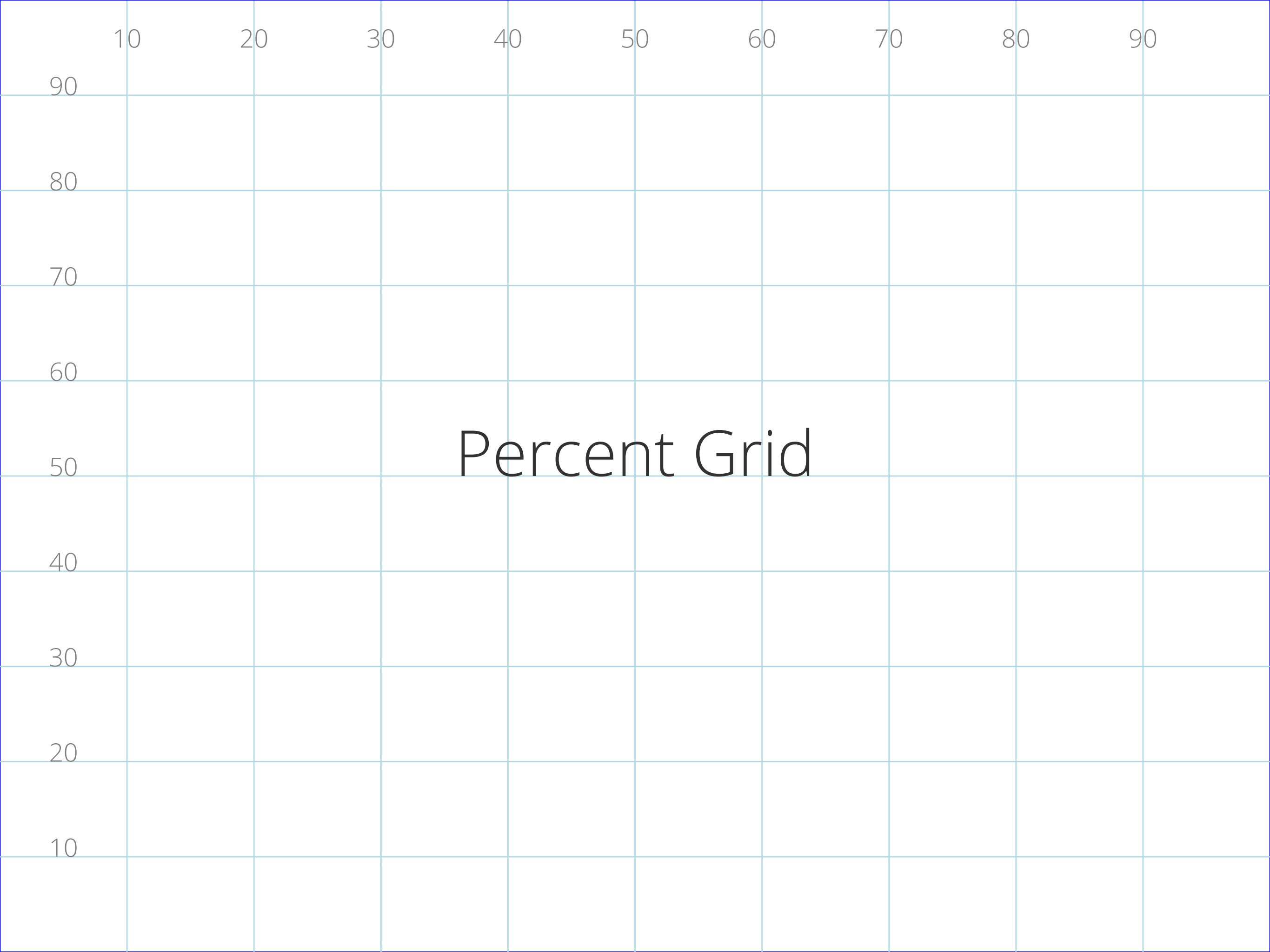
```
<ellipse xp="10" yp="35" wp="15" hr="66.66" color="green"/>
```

```
<ellipse xp="15" yp="30" wp="10" hr="100" color="blue" opacity="30"/>
```



```
<curve xp1="5" yp1="10" xp2="15" yp2="20" xp3="15" yp3="10" sp="0.3" color="red"/>
```

```
<arc xp="22" yp="10" wp="10" wp="10" a1="0" a2="180" sp="0.2" color="blue"/>
```



Percent Grid

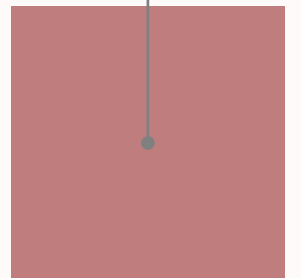
10%, 50%

Hello

50%, 50%



90%, 50%



Percentage-based layout

bullet

- Point A
- Point B
- Point C
- Point D

plain

First item
Second item
The third item
and the last thing

number

1. This
2. That
3. The other
4. One more

```
<list>...</list>
```

bullet

- Point A
- Point B
- Point C
- Point D

plain

First item
Second item
The third item
and the last thing

number

1. This
2. That
3. The other
4. One more

```
<list>...</list>
```

Design Examples

Top

Left

Right

Bottom

20%

30%

70%

20%

Header (top 20%)

Summary
(30%)

Detail
(70%)

Footer (bottom 20%)

Two Columns

One

Two

Three

Four



Tree and Sky

Five

Six

Seven

Eight



Rocks

This is not a notecard

Rich

Can't buy me love

Bliss

Worse

Better

Misery

We have each other

Poor

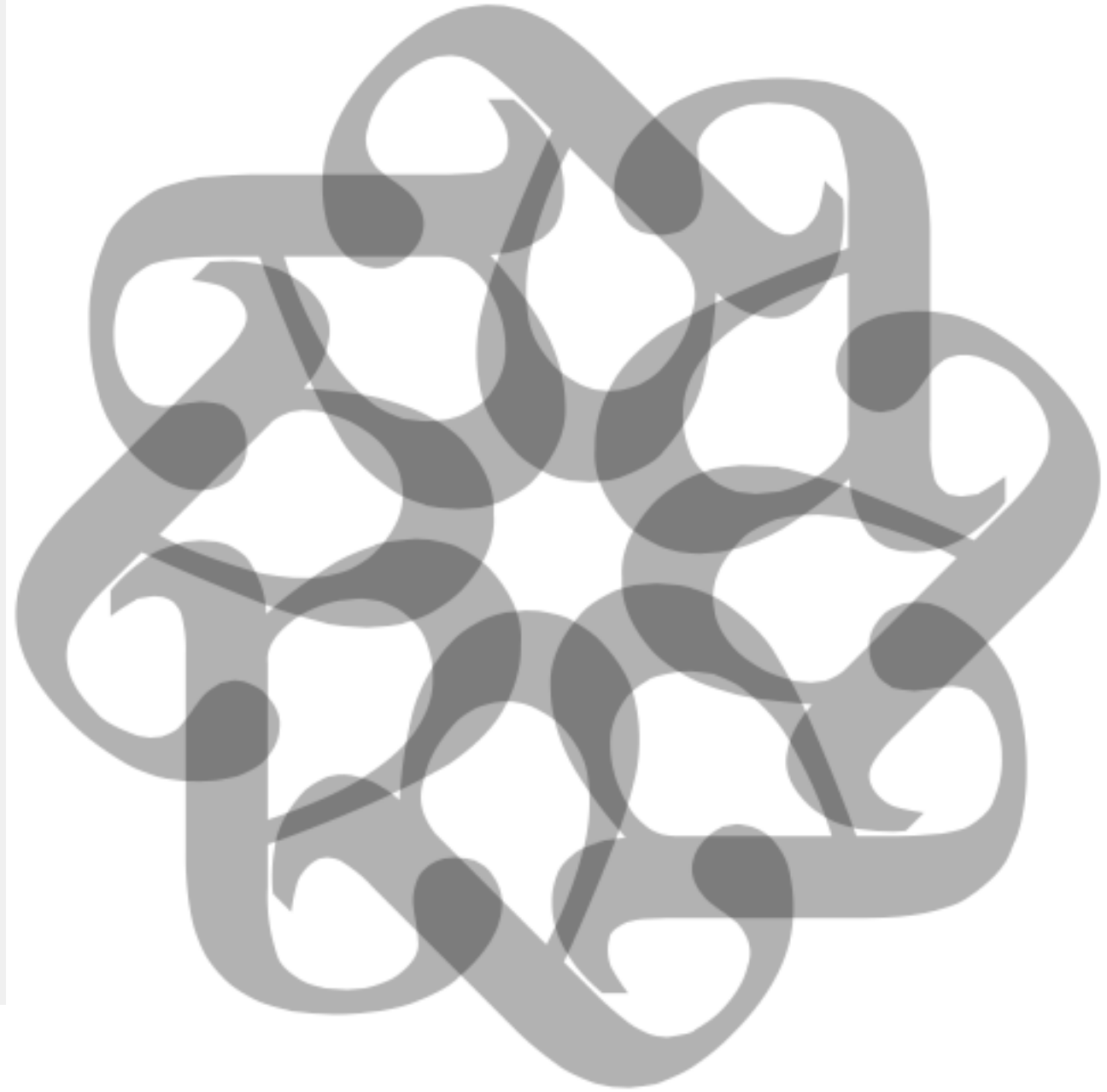
Code

```
package main

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

func main() {
    canvas := svg.New(os.Stdout)
    width := 850
    height := 1100
    canvas.Start(width, height)
    canvas.Gstyle("fill-opacity:0.3;font-size:480pt")
    for r := 0.0; r < 360.0; r += 45 {
        canvas.TranslateRotate(width/2, height/2, r)
        canvas.Text(0, 0, "a")
        canvas.Gend()
    }
    canvas.Gend()
    canvas.End()
}
```

Output



A few months ago, I had a look at the brainchild of a few serious heavyweights working at Google. Their project, the Go programming language, is a static typed, c lookalike, semicolon-less, self formatting, package managed, object oriented, easily paralellizable, cluster fuck of genius with an unique class inheritance system. It doesn't have one.

The Go Programming Language

is a static typed,
c lookalike,
semicolon-less,
self formatting,
package managed,
object oriented,
easily paralellizable,
cluster fuck of genius
with an unique class inheritance system.

The Go Programming Language

is a static typed,
c lookalike,
semicolon-less,
self formatting,
package managed,
object oriented,
easily paralellizable,
cluster fuck of genius
with an unique class inheritance system.

The Go Programming Language

is a static typed, c lookalike, semicolon-less, self formatting,
package managed, object oriented, easily paralellizable,
cluster fuck of genius with an unique class inheritance system.

It doesn't have one.

So, the next time you're about
to make a subclass, think hard
and ask yourself

what would Go do

Andrew Mackenzie-Ross, <http://pocket.co/sSc56>





You must not blame me if I do talk to the clouds.

FOR, LO,

the winter is past,

the rain is over and gone;

The flowers appear on the earth;

the time for the singing of birds is come,

and the voice of the turtle is heard in our land.



Clients

```
package main

import (
    "log"
    "github.com/ajstarks/deck"
)

func main() {
    presentation, err := deck.Read("deck.xml", 1024, 768) // open the deck
    if err != nil {
        log.Fatal(err)
    }
    for _, slide := range presentation.Slide {           // for every slide...
        for _, t := range slide.Text {                   // process the text elements
            x, y, size := deck.Dimen(presentation.Canvas, t.Xp, t.Yp, t.Sp)
            slideText(x, y, size, t)
        }
        for _, l := range slide.List {                   // process the list elements
            x, y, size := deck.Dimen(presentation.Canvas, l.Xp, l.Yp, l.Sp)
            slideList(x, y, size, l)
        }
    }
}
```

A Deck Client

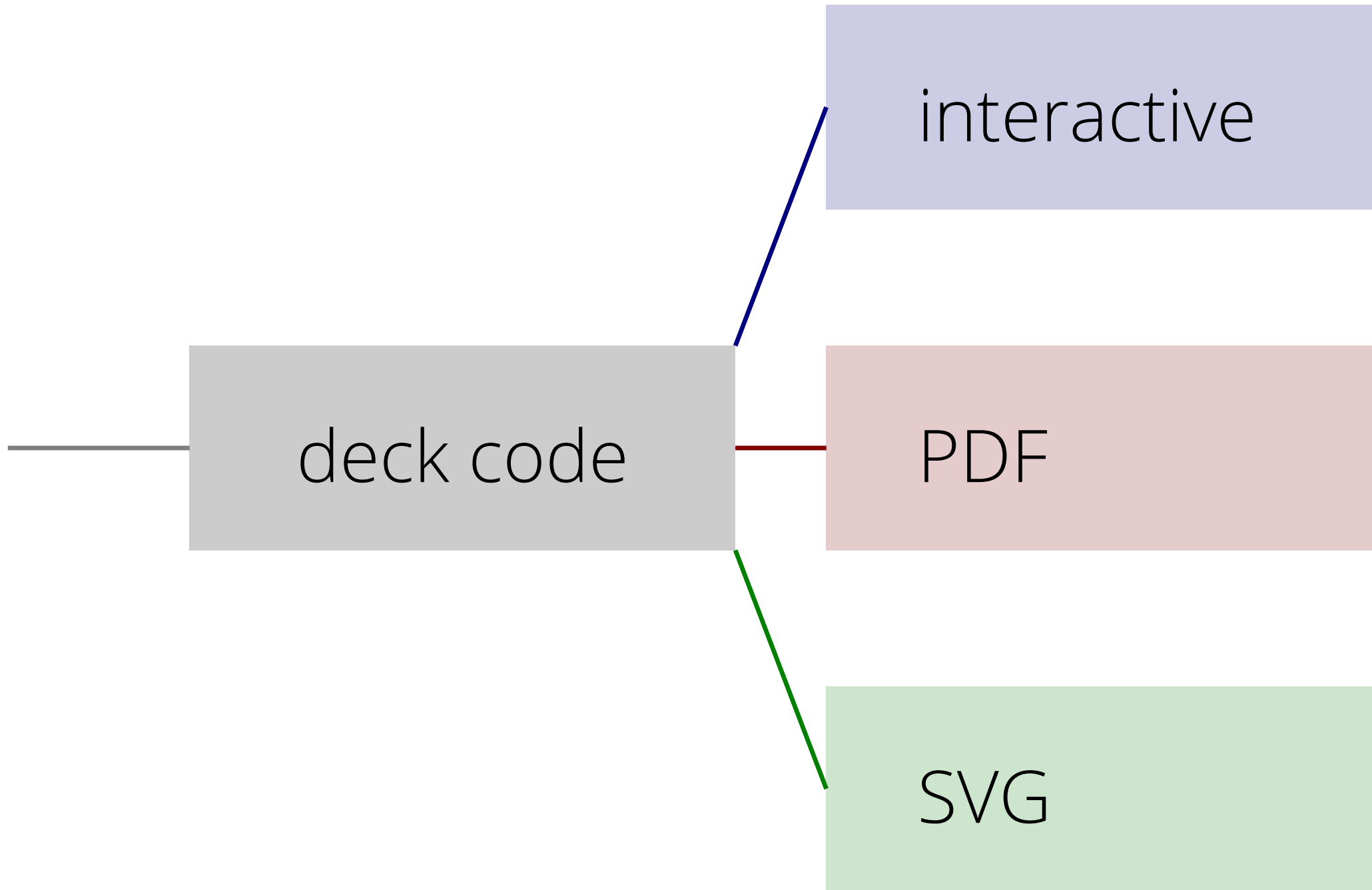
Process

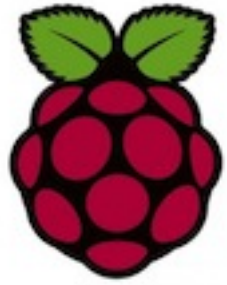
deck code

interactive

PDF

SVG





```
go get github.com/ajstarks/deck/vgdeck
```



```
go get github.com/ajstarks/deck/pdfdeck
```



```
go get github.com/ajstarks/deck/svgdeck
```


pdfdeck [options] file.xml...

- sans, -serif, -mono [font] specify fonts
- pagesize [Letter, Legal, Tabloid, A2, A3, A4, A5, ArchA, Index, 4R, Widescreen]
- pagewidth [page width (pt)]
- pageheight [page height (pt)]
- stdout (output to standard out)
- outdir [directory] directory for PDF output
- fontdir [directory] directory containing font information
- author [author name] set the document author
- title [title text] set the document title
- grid [percent] draw a percent grid on each slide

svgdeck [options] file.xml...

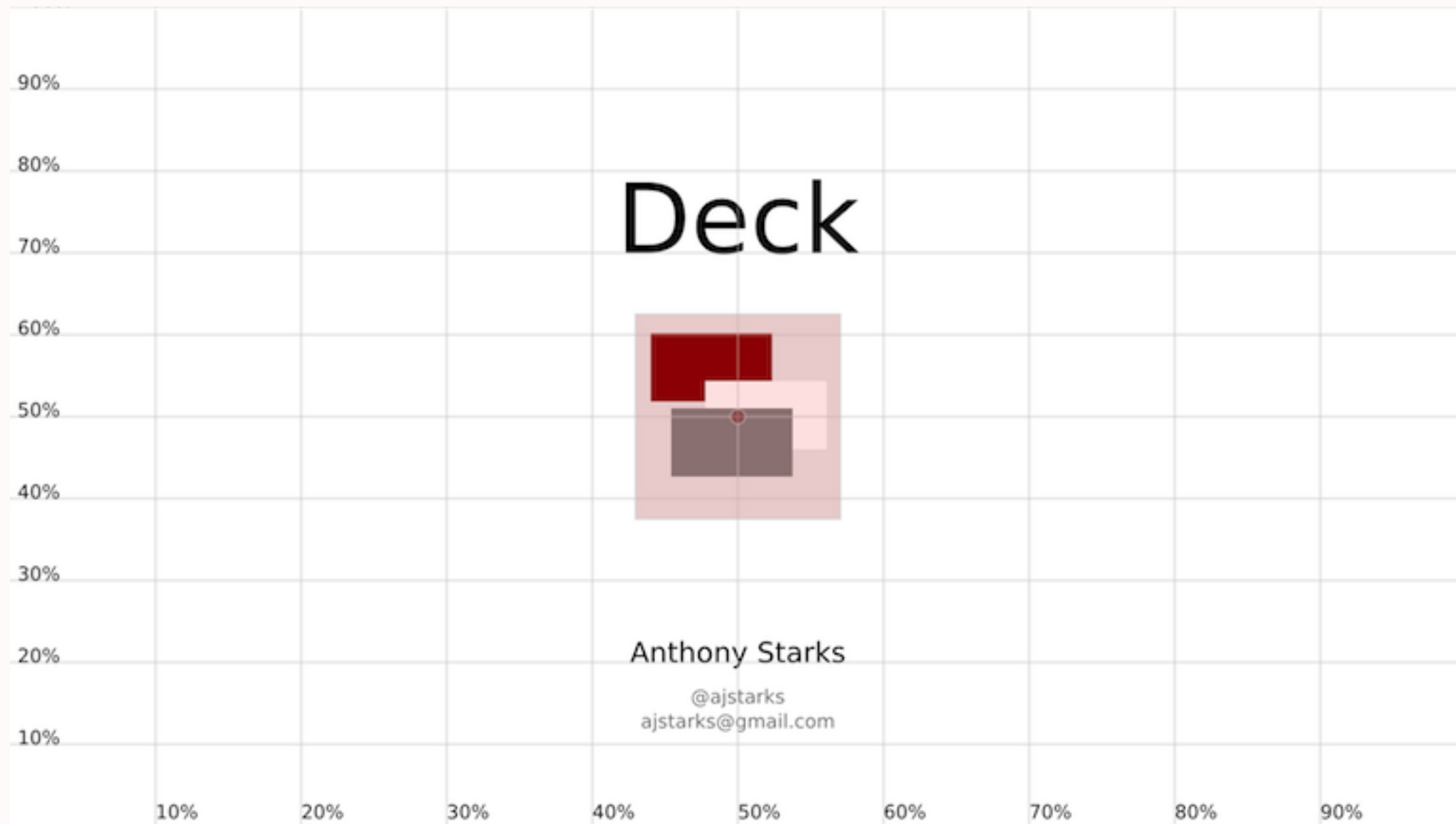
- sans, -serif, -mono [font] specify fonts
- pagesize [Letter, Legal, A3, A4, A5]
- pagewidth [canvas width]
- pageheight [canvas height]
- stdout (output to standard out)
- outdir [directory] directory for PDF output
- title [title text] set the document title
- grid [percent] draw a percent grid on each slide

vgdeck [options] file.xml...

- loop [duration] loop, pausing [duration] between slides
- slide [number] start at slide number
- w [width] canvas width
- h [height] canvas height
- g [percent] draw a percent grid

vgdeck Commands

<code>+</code> , <code>Ctrl-N</code> , <code>[Return]</code>	Next slide
<code>-</code> , <code>Ctrl-P</code> , <code>[Backspace]</code>	Previous slide
<code>^</code> , <code>Ctrl-A</code>	First slide
<code>\$</code> , <code>Ctrl-E</code>	Last slide
<code>r</code> , <code>Ctrl-R</code>	Reload
<code>x</code> , <code>Ctrl-X</code>	X-Ray
<code>/</code> , <code>Ctrl-F</code> <code>[text]</code>	Search
<code>s</code> , <code>Ctrl-S</code>	Save
<code>q</code>	Quit



X-Ray mode shows the percent grid, and highlights images

github.com/ajstarks/deck



ajstarks@gmail.com

@ajstarks