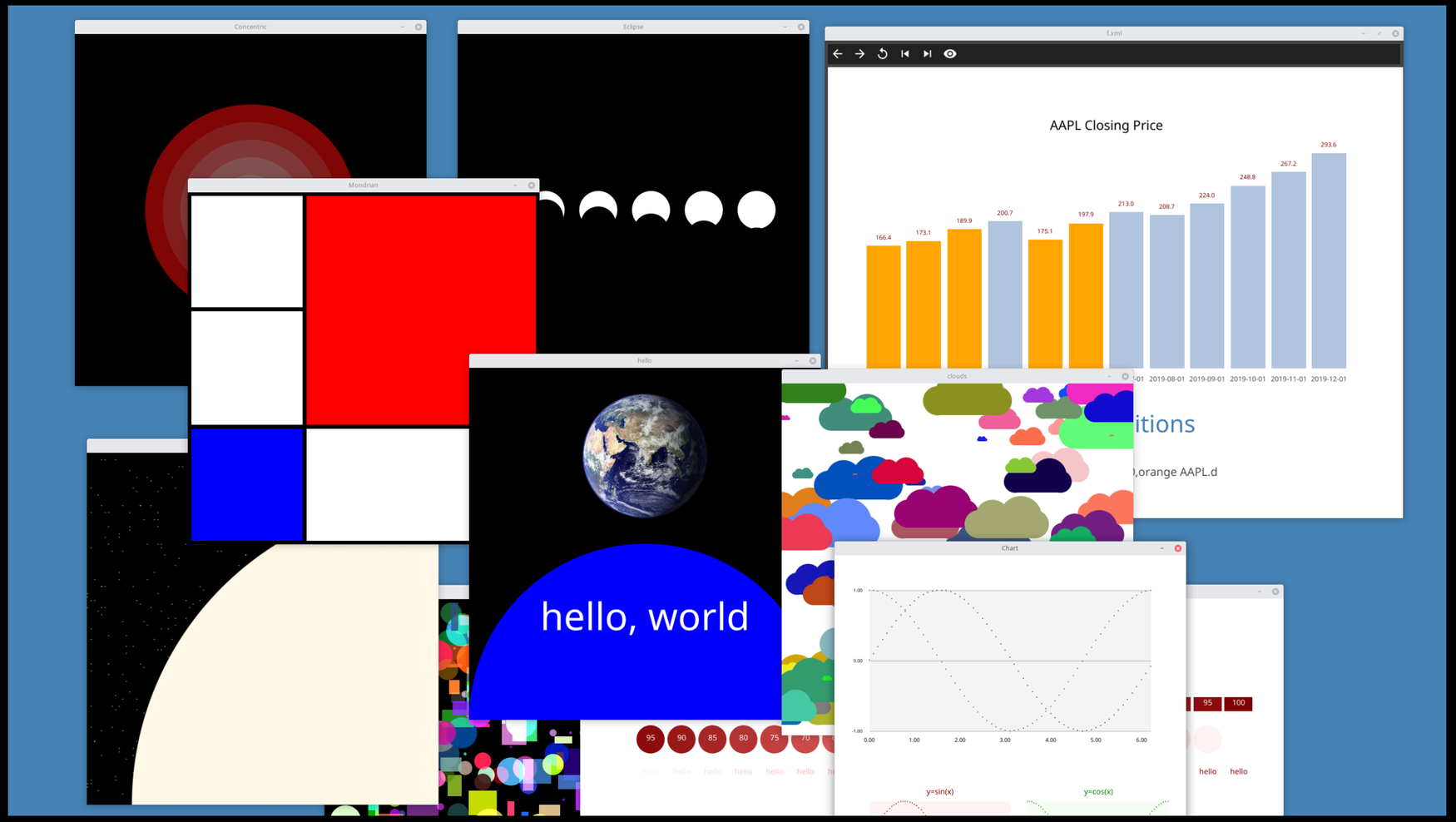
fc a high-level canvas API for the fyne toolkit



Anthony Starks @ajstarks

Elements

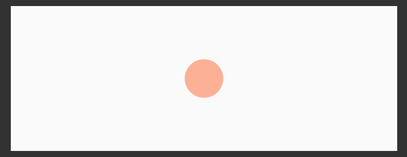
Text

CText

EText

circle

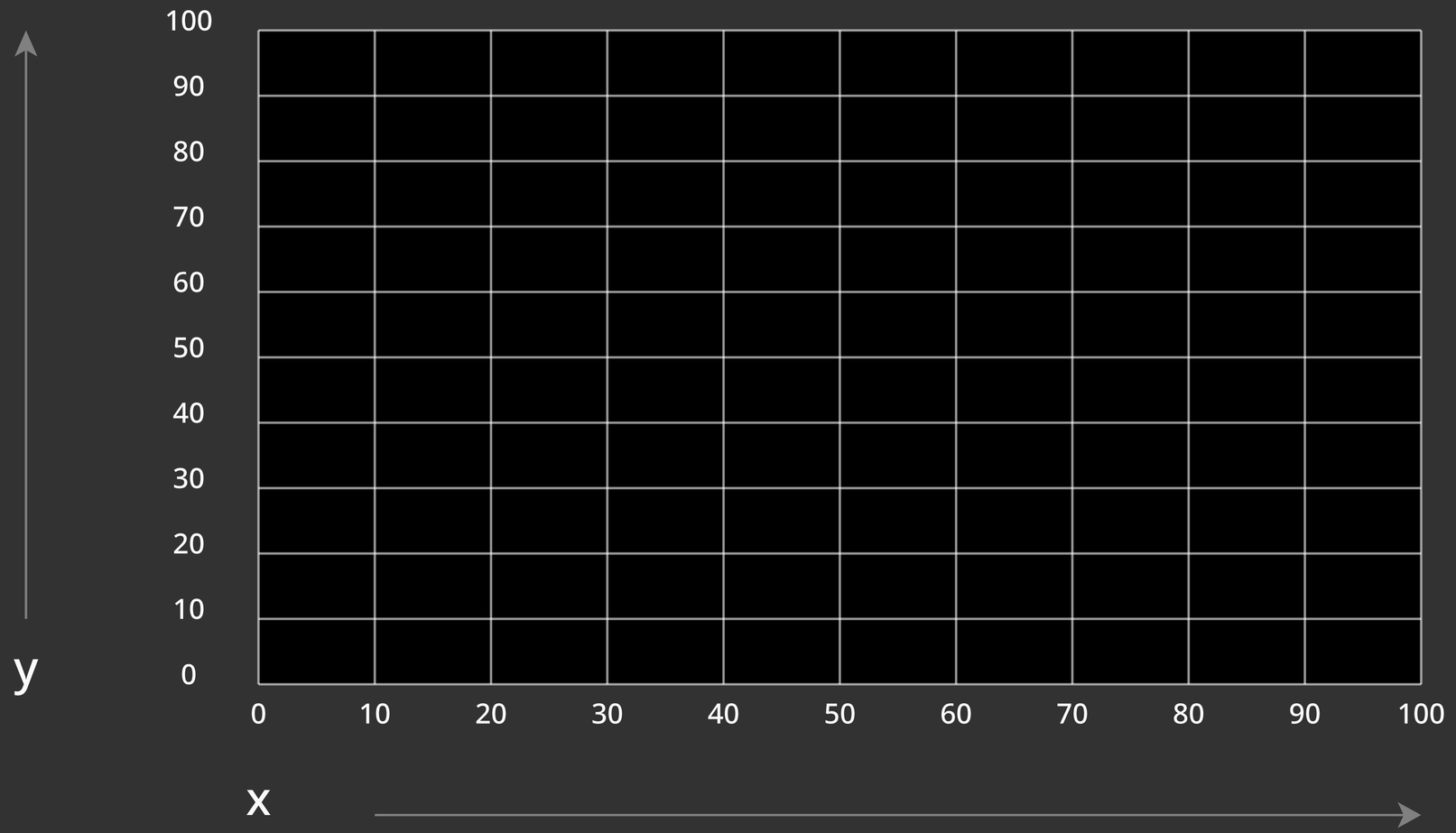
line



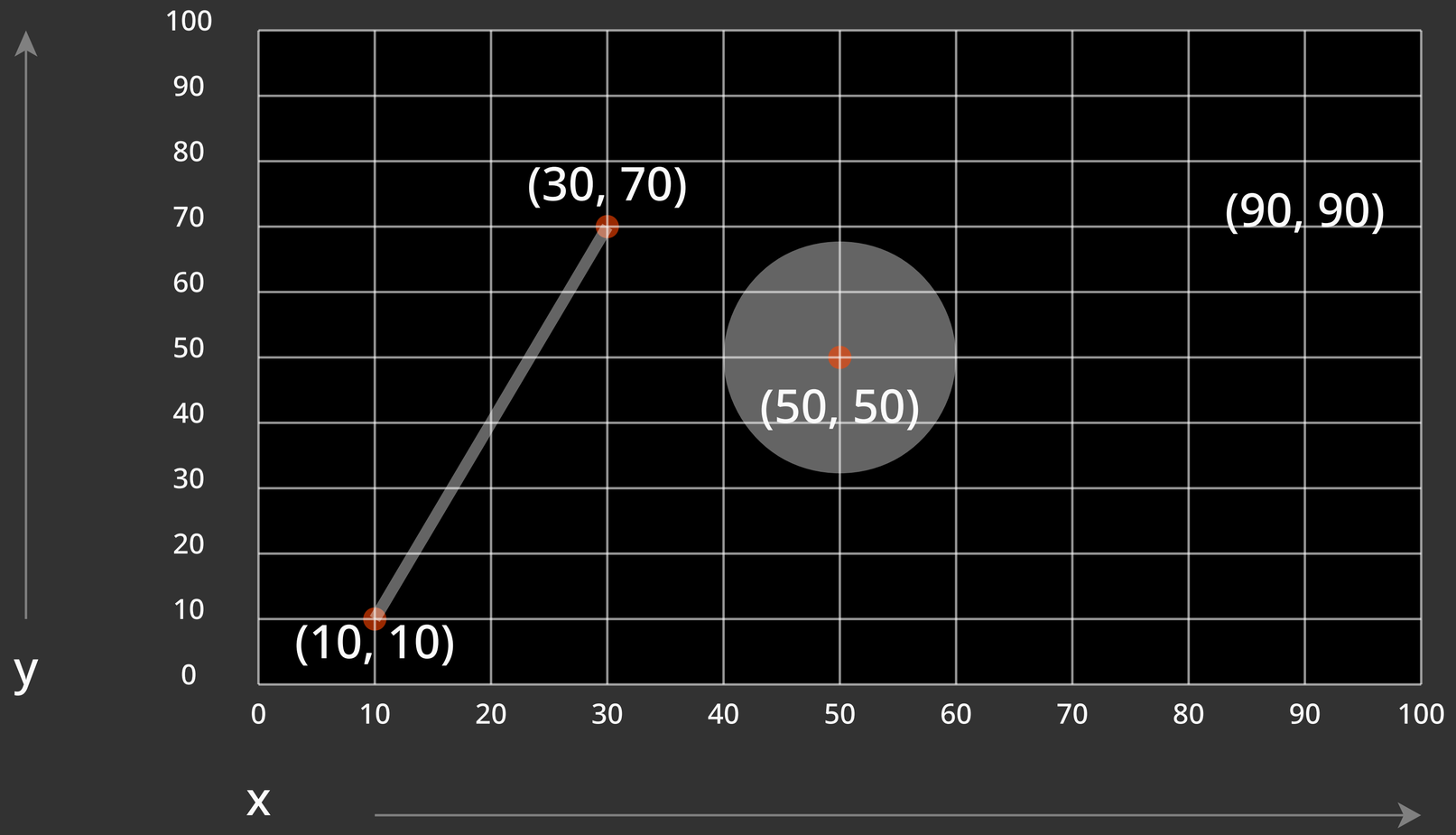
rectangle

image

Percentage-based Grid



Using the Percentage-based Grid



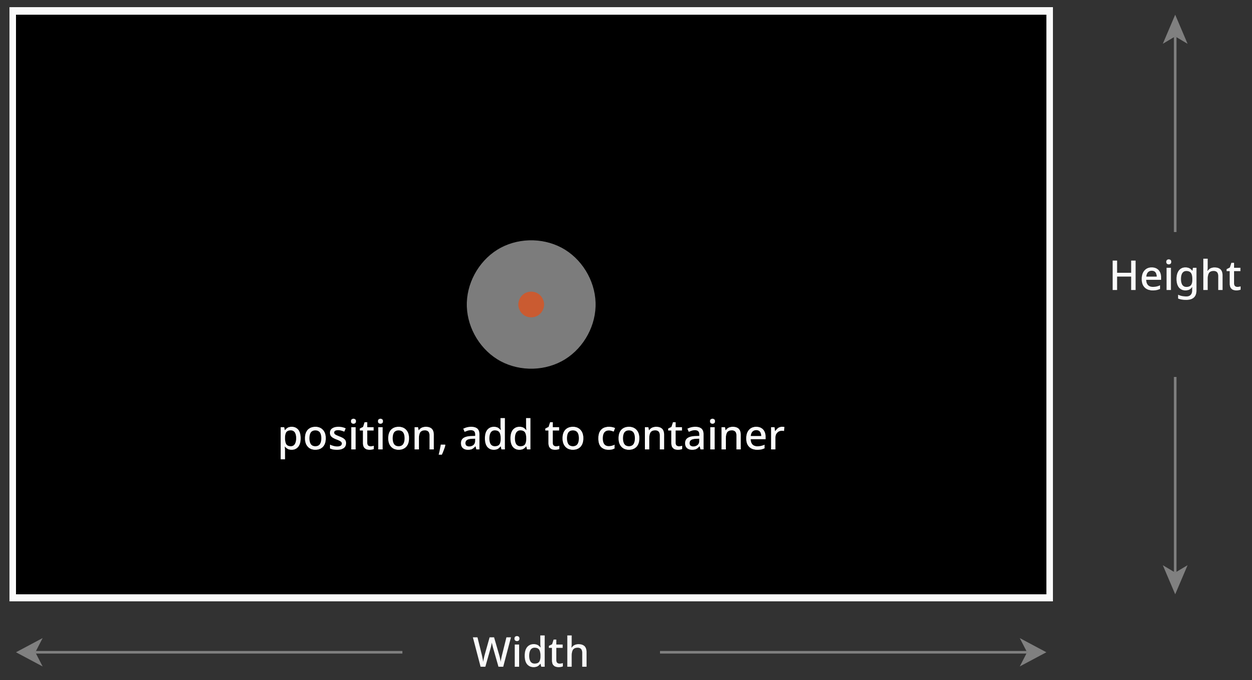
Rect(90, 80, ...)

Circle(50, 50, ...)

Line(10, 10, 30, 70, ...)

fc structures

```
// Canvas is where objects are drawn into
type Canvas struct {
    Window    fyne.Window
    Container *fyne.Container
    Width     float64
    Height    float64
}
```



Percentage-based Methods on *Canvas

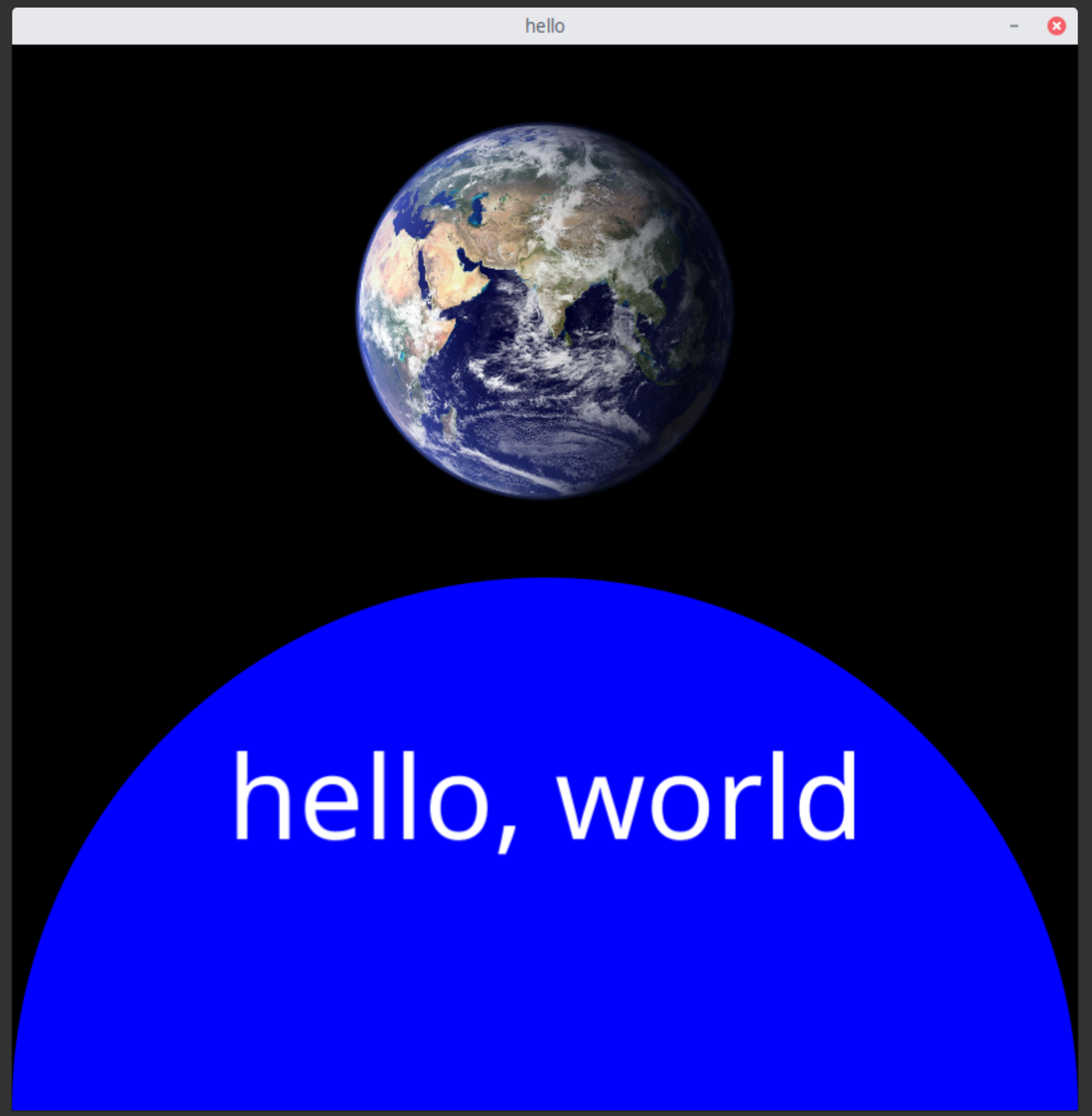
Make a new canvas	`NewCanvas(name string, w, h int) Canvas`
Place text, left-aligned	`Text(x, y, size float64, s string, fill color.RGBA)`
Place centered text	`CText(x, y, size float64, s string, fill color.RGBA)`
Place end-aligned text	`EText(x, y, size float64, s string, fill color.RGBA)`
Obtain the text width	`TextWidth(s string, size float64) float64`
Circle centered (x,y), radius r	`Circle(x, y, r float64, fill color.RGBA)`
Rectangle, upper-left at (x,y)	`CornerRect(x, y, w, h float64, fill color.RGBA)`
Rectangle centered at (x,y)	`Rect(x, y, w, h float64, fill color.RGBA)`
Line from (x1,y) to (x2,y2)	`Line(x1, y1, x2, y2, size float64, stroke color.RGBA)`
Image centered at (x,y)	`Image(x, y float64, w, h int, name string)`
Display and run	`EndRun()`

Convenience methods

Lookup colors by name	`ColorLookup(s string) color.RGBA`
Map one range into another	`MapRange(value, low1, high1, low2, high2 float64) float64`
Polar to Cartesian coordinates	`Polar(x, y, r, angle float64) (float64, float64)`
Convert degrees to radians	`Radians(deg float64) float64`

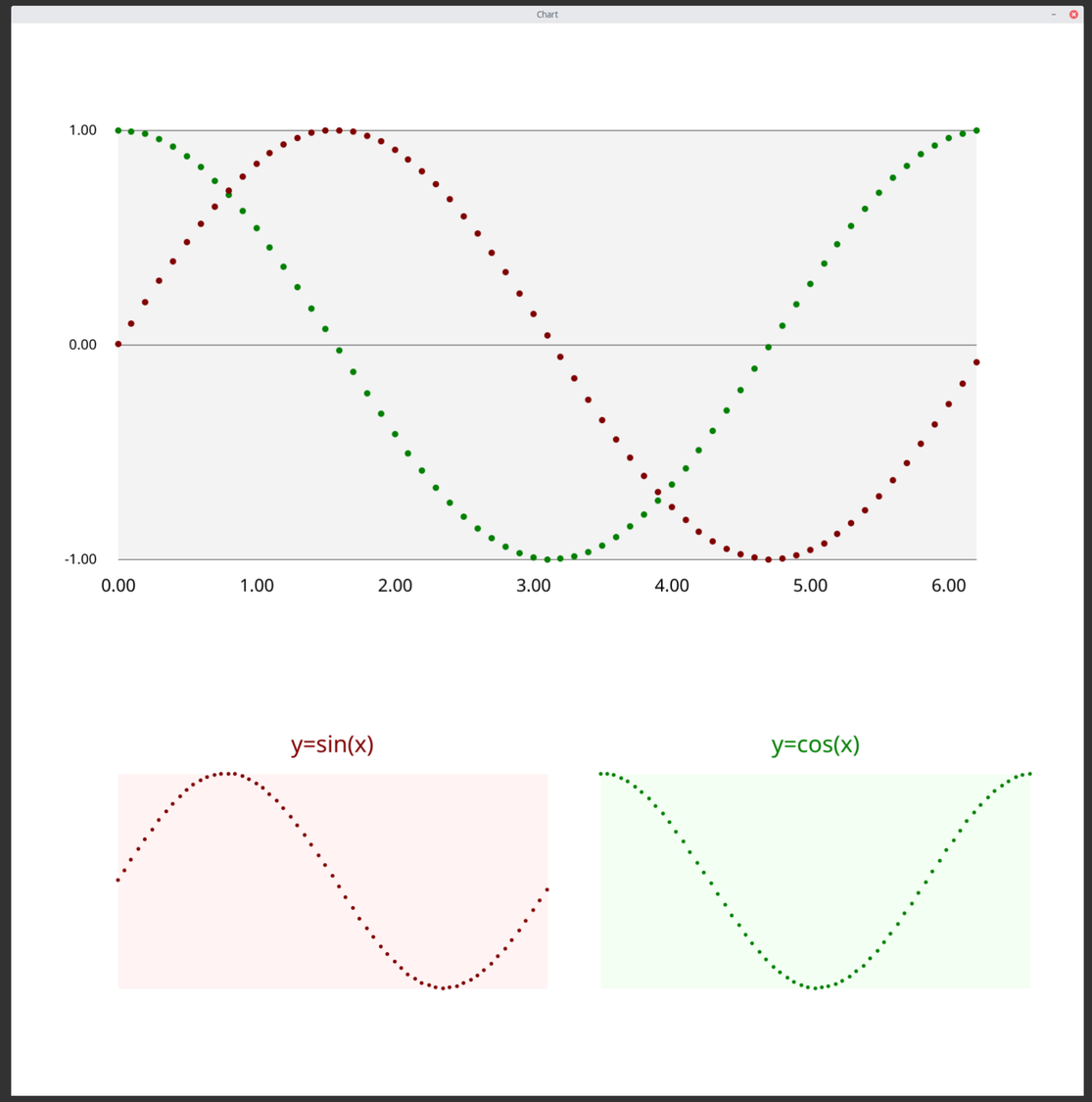
hello, (fc) world

```
package main

import (
    "image/color"

    "github.com/ajstarks/fc"
)

func main() {
    width, height := 500, 500
    blue := color.RGBA{0, 0, 255, 255}
    white := color.RGBA{255, 255, 255, 255}

    canvas := fc.NewCanvas("hello", width, height)

    canvas.Circle(50, 0, 100, blue)
    canvas.CText(50, 25, 10, "hello, world", white)
    canvas.Image(50, 75, 200, 200, "earth.jpg")

    canvas.EndRun()
}
```



fc/chart:

```
cosine.Frame(canvas, 5)
cosine.Label(canvas, 1.5, 10)
cosine.YAxis(canvas, 1.2, -1.0, 1.0, 1.0, "%0.2f", true)
cosine.Color = color.RGBA{0, 128, 0, 255}
sine.Color = color.RGBA{128, 0, 0, 255}
cosine.Scatter(canvas, 0.75)
sine.Scatter(canvas, 0.75)

sine.Left = 10
sine.Right = sine.Left + 40
sine.Top, cosine.Top = 30, 30
sine.Bottom, cosine.Bottom = 10, 10

sine.CTitle(canvas, 2, 2)
sine.Frame(canvas, 5)
sine.Scatter(canvas, 0.5)

offset := 45.0
cosine.Left = sine.Left + offset
cosine.Right = sine.Right + offset

cosine.CTitle(canvas, 2, 2)
cosine.Frame(canvas, 5)
cosine.Scatter(canvas, 0.5)
```



fc/chart: data structures

```
// NameValue is a name,value pair
type NameValue struct {
    label string
    note  string
    value float64
}

// ChartBox holds the essential data for making a chart
type ChartBox struct {
    Title                   string
    Data                    []NameValue
    Color                   color.RGBA
    Top, Bottom, Left, Right float64
    Minvalue, Maxvalue      float64
    Zerobased               bool
}
```

fc/chart methods on *ChartBox

Read data, make a ChartBox	`func DataRead(r io.Reader) (ChartBox, error)`
Bar Chart	`Bar(canvas fc.Canvas, size float64)`
Horizontal Bar Chart	`HBar(canvas fc.Canvas, size, linespacing, textsize float64)`
Line Chart	`Line(canvas fc.Canvas, size float64)`
Scatter Chart	`Scatter(canvas fc.Canvas, size float64)`
Centered Title	`CTitle(canvas fc.Canvas, size, offset float64)`
Chart Frame	`Frame(canvas fc.Canvas, opacity float64)`
X Axis Label	`Label(canvas fc.Canvas, size float64, interval int)`
Y axis	`YAxis(canvas fc.Canvas, size, min, max, step float64, fmt string, grid bool)`