CS 341: Project 3—Go Interfaces (rev. 1)

Jon A. Solworth

Due: Dec. 1 2023 at 11:59 PM

1 Note

This document supersedes the slides presented in class and the earlier project description. This is the first revision, available on 21 November.

2 Problem

2.1 Description

You are to create two interfaces, `geometry` and `screen`, and a color map. Your program will draw the `geometry` (either `Rectangle`, `Circle`, or `Triangle`) on the `screen` (implemented as a logical device of type `Display`).

2.2 Color map

The color map, `cmap`, maps from `Color`s (ints) to `r,g,b`. Note that `cmap` need not be implemented as a Go `map`. The `cmap` should have colors for `red`, `green`, `blue`, `yellow`, `orange`, `purple`, `brown`, `black`, `white`.

Use the following colors:

color	**R**	**G**	**B**
red	255	0	0
green	0	255	0
blue	0	0	255
yellow	255	255	0
orange	255	164	0
purple	128	0	128
brown	165	42	42
black	0	0	0
white	255	255	255

2.3 Geometry

There are three object types which fit the `geometry` interface are:

Rectangle specified by a

- `ll` lower-left hand corner
- `ur` upper-right hand corner
- `c` the color

Circle specified by a

- `cp` center point
- `r` radius
- `c` color

Triangle triangle specified by three points (corners)

- `pt1, pt2, pt3` the three points of the triangle
- `c` the color

Associated with the `geometry` interface are the methods:

- `draw(scn screen) (err error)` draw the filled-in shape on the screen
- `shape() (s string)` print the type of the object

2.4 Screen

Associated with each `screen` interface is

- `initialize(maxX, maxY int)` initialize a screen of `maxX` times `maxY`. Called before any other method in screen interface. Clears the screen so that it is all white background.
- `getMaxXY() (maxX, maxY int)` get the `maxX, maxY` dimensions of the screen
- `drawPixel(x, y int, c Color) (err error)` Draw the pixel with color `c` at location `x,y`
- `getPixel(x, y int) (c Color, err error)` Get the color of the pixel at location `x,y`
- `clearScreen()` clear the whole screen by setting each pixel to `white`
- `screenShot(f string) (err error)` dump the screen as a `f.ppm` as in project 2. Note that you must reproduce the form exactly; code was provided in project 2, so you can use that to generate a sample output. You can then view the graphics output with an image viewer (I used eog on Linux). Error if the file cannot be written.

2.5 Display

A screen is an interface. The underlying type could be a logical device (i.e. data structure) or a physical display (such as a monitor). We describe here a logical device, which is initialized with the dimensions of the display and a matrix created with a slice of slices which contains the current color at each pixel in the display. The Display's pixels are initialized to be white.

```
struct Display {
    maxX, maxY int
    matrix [][]Color
}
```

The members maxX (respectively maxY) is the size of the screen. For example, a 1024x1024 has valid x (resp. y) between 0 and 1023.

3 Go programming issues

3.1 packages

Use only the packages fmt, math, errors and os.

3.2 geometry code

The project is not about the geometry, its about the programming language. Therefore you can take code from the Internet to draw the Circle, Triangle, and Line (used to draw the interior of the geometry). You should document the source of whatever code you take with a URL.

Note that you should write the code for Rectangle yourself.

The starter code contains draw code for the Triangle.

This is the only exception to the plagiarism rule for the course.

3.3 Errors

You should detect whether a Color is illegal or whether the figure is drawn outside the screen. In either case you should return an error from draw, not draw anything on the screen, and print a suitable error message. In addition, getPixel and drawPixel return errors.

The errors package will enable you to create new errors.

4 Example

4.1 main()

Here is an incomplete main example. Feel free to extend it for full testing.

```
func main() {
        fmt.Println("starting␣...")
        display.initialize(1024,1024)

        rect :=  Rectangle{Point{100,300}, Point{600,900}, red}
        err := rect.draw(&display)
        if err != nil {
                fmt.Println("rect:␣", err)
        }

        rect2 := Rectangle{Point{0,0}, Point{100, 1024}, green}
        err = rect2.draw(&display)
        if err != nil {
                fmt.Println("rect2:␣", err)
        }

        rect3 := Rectangle{Point{0,0}, Point{100, 1022}, 102}
        err = rect3.draw(&display)
        if err != nil {
                fmt.Println("rect3:␣", err)
        }

        circ := Circle{Point{500,500}, 200, green}
        err = circ.draw(&display)
        if err != nil {
                fmt.Println("circ:␣", err)
        }

        tri := Triangle{Point{100, 100}, Point{600, 300},  Point{859,850}, yellow}
        err = tri.draw(&display)
        if err != nil {
                fmt.Println("tri:␣", err)
        }

        display.screenShot("output")
}
```

4.2 Graphics output

Figure 1 shows the graphic output for the example main given earlier.

4.3 Text output

This is the text output:

```
starting ...
rect2:  geometry out of bounds
rect3:  color unknown
```

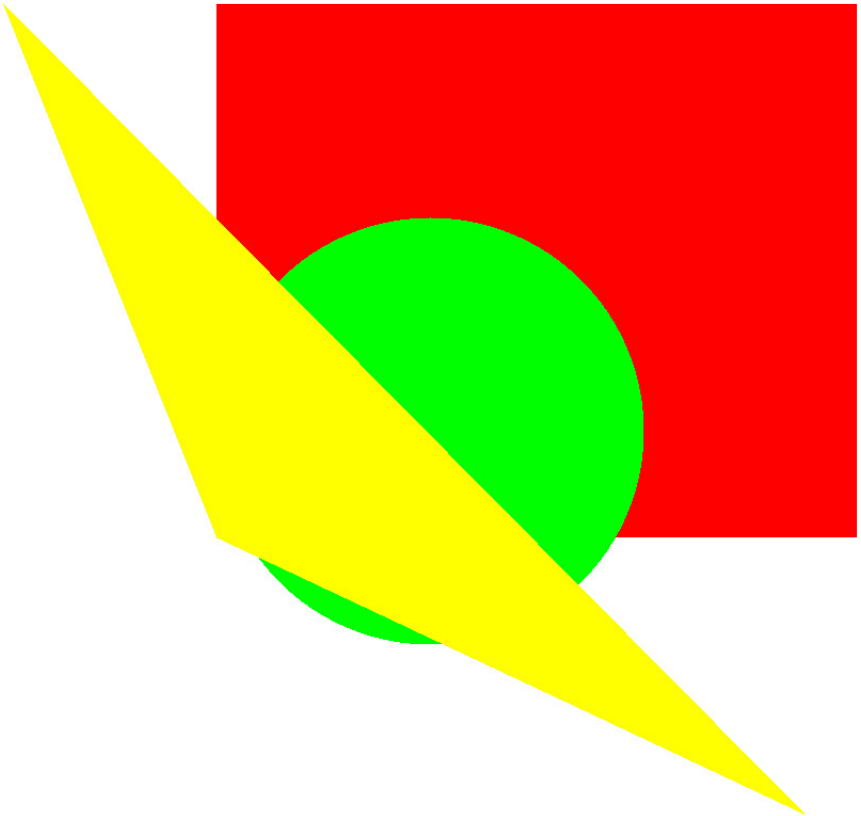

Figure 1: Here is the output for the main() shown above

5 Palette

In Figure 2 is the output of printing all the shapes in all the colors.

6 References

triangles `https://gabrielgambetta.com/computer-graphics-from-scratch/07-filled-triangles.html`

circles `https://www.redblobgames.com/grids/circle-drawing`

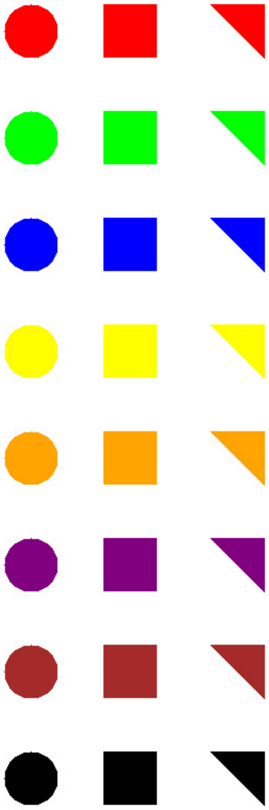

Figure 2: Here is the a palette containing all the shapes and colors (can't see white)