Project 1 - Image Server

In this project you will create a drawing eDSL (you can take inspiration in your design from the Shape language we used in the lectures, or from the Pan language, or neither). You will combine it with the two web eDSL languages we saw (Scotty and Blaze) to produce a web application capable of delivering images. The web app does **not** need to be interactive (that is, the images can be hard-coded in the app), the point is to create a DSL that could be used to specify images and integrate it into an application.

You will use the JuicyPixels library to create the images (you will receive sample code for this. The library can serve as an almost drop-in replacement for the Ascii rendering layer in the example code, though there are also more efficient ways to make use of it).

This project is overall worth 25% (the second project will be worth 35%). The project is due at midnight on the 7th of November - that's 4 weeks from the day of release. Any extensions to the deadline must be agreed in advance, in writing.

The project tasks, in detail, are:

- Design a suitable drawing eDSL and an interpretation function that renders drawings using JuicyPixels [60% of the project marks for this].
    - Provide at least the following basic shapes: Circle, Rectangle, Ellipse, Polygon (this last should be a closed convex polygon defined by a series of points listed in clockwise order).
    - Provide the following set of basic affine transformations: Scale, Rotate, Shear, Translate, and functions to combine the transformations with shapes.
    - Provide a way to specify colour for each shape. This can be either a simple case with a single colour, or a function which can provide gradients.
    - Provide a way to join two drawings together with relative positioning, as described in the shape
- As a UI provide a Scotty application which can render some (hard-coded) sample images that demonstrate the result. The images should be returned as *PNG graphics* rendered using JuicyPixels. You should include the text of the DSL program that produced the image in the web page, so that the user of the web app can see how the image was produced (the idea is that a future improvement could be to allow the user to edit this text and re-render it, but you don't have to implement that). [20% of the project marks for this]
- Document your solution. [20% of the project marks for this]

Required features of the language:

- The drawing language must support at least the following basic shapes: Circle, Rectangle, Ellipse, Polygon (this last should be a closed convex polygon defined by a series of points listed in clockwise order).
- It should be possible to transform each shape using the following set of basic affine transformations: Scale, Rotate, Shear, Translate.

- It should be possible to specify the *colour* of each shape (we will not concern ourselves with outlines and strokes, etc. You may assume the colour is just the solid fill colour of the interior of the shape)
- A drawing is either a single shape, or a shape *combined with another drawing* using one of the following operations:
    - "over": the first shape is drawn directly over the second drawing - for example a circle 'over' a square would look like a square with a circle inscribed in it.
    - "left", "right", "below", "above": the first shape is drawn beside the second drawing so that it does not overlap it at all, but it placed to one of the four sides.

Submit

- Your final program **as a stack project**, including all the source files needed to build and run it.
- A short document (text or PDF) outlining the design choices you made, and indicating how many of the project deliverables were satisfactorily completed. Include a section for each of the project tasks, and for your optimisation explain what you have attempted.
  Include in this document a reflection on the process of designing the DSL; some examples of questions you might address include: did you find the design choices you had to make challenging? At any point did you have to revise your language design? How well did your choices fit with the needs to the rendering library chosen, and how might they have changed if the final rendering target was instead an SVG produced by Blaze? The reflection can be short (one or two paragraphs).

Update

I had some questions about what the report needs to contain and how long it should be.

1. The report does *not* need to include lots of code -- it will be read in conjunction with your code submission, so you can make reference to function names, lines of code, etc. from the source

2. The report need only be 2-3 pages long. You will probably need more than one page to address all the points, but it is not necessary to write a very long report here but it should be addressing design and overview questions that you might not include in the comments.