Simple Fun Fourier Drawing Programm

Inspiration

- 3Blue1Brwon's amazing **video** about fourier series
- Fourier Series can draw anything!
- In the form of Integral, the series will approach the oringinal image as the number of terms rise

$$f(t) = \int_0^1 c_n e^{2i\pi n*t} dt$$

- The Mathematical Pinciple behind the image is just amazing

Data Format

- prepare your data in the following json format, where each object of series represents a term of fourier series

```
{
    "series": [
        {
            "n" : 0,
            "real" : 1,
            "imag" : 2
        }
    ]
}
```

- `n` represents exactly n in the form, which should be a Integer, and controls how fast the vector spins
- `real` and `imag` represent the complex constant c_n in the form, control the starting position and length of the vector
- Then put them in one directory without other files. Each file will be transformed to a closed curve. In order to draw a image with multiple curves, you'll need the same amount of json files.

Usage

- Use existing `Makefile`
  - `make` : build the executables
  - `make clean` : clean the generated executables/artifacts
  - `make doc` : generate a pdf versoin of the document
  - `make png` : use the json file in directory `data` to draw the whole image and save the output in directory `out`
  - `make gif` : render the drawing process as an animation and save the output in directory `out`
    - warnging : too many frames may cause a horrible runtime which would take hours to render

  - `make evo` : render the evolution process as the number of vectors rises for each image
- Use executable directly
  - `Main <filetype> <source path> <output path>`
  - filetype : `png`, `gif` or `evo`
  - source path : the path where json file exists
  - output path : the path where images are saved to

Customization

- At the beginning of the source code are several option that can be customized easily
  - `width`, `height` : the height and width of the generated image
  - `white`, `black` : the standard pixel color
  - `pixelArt` : the function that dertermines color and by default draws black
  - `pointCount` : the number of points to be painted
  - `gifStep` : how many pixels will be rendered each frame
    * warning : terrible runtime if set too small
  - `filename` : the name of outputed image
  - `scaleFactor` : how much the final image will be scaled

Dependencies

- **aeson** : for json input parsing
- **JuicyPixels** : for pixel drawing
- **attoparsec** : for some weird functions
- **pandoc** : to convert this manual to pdf
- **juicy-draw** : for line drawing (not included in the final version)

Known Limits

- Due to lack of time I didn't implemented the part where original image are converted to Fourier Series
  and instead just used 3Blue1Brown's functions to calculate the series link

- Also due to lack of time I didn't optimized the gif generation algorithm, which has a terrible runtime when there's too many points. The optimization seems eazy but I had a real head ache on understanding the state monad XD. It took me around 10 hours to render the whole project, so **Do Not Try At Home**, unless you have plenty of computing horse power and time

- Didn't draw the vectors in the gif

- Should have been drawing vectorgraph from the very beginning.
  bitmap images are just too slow in this case