Millipeds Git Repo

Program Explanation

Date April 11, 2023

Contents

Pixel

A pixel data structure in this program is defined as follows:

```
1  typedef struct PIXEL_T {
2    int r;
3    int g;
4    int b;
5  } * pixel;
```

And can be understood as a collection:

$$\text{Pixel} = (r, g, b)$$

Such that r, g, and b are signed integers.

Canvas

A canvas data structure in this program is defined as:

```
1  typedef struct CANVAS_T {
2    pixel ** values;
3    int height;
4    int width;
5  } * canvas;
```

From this definition we can think of a canvas as a matrix:

$$\text{Canvas} = \begin{bmatrix} \text{Pixel}_{11} & \dots & \text{Pixel}_{1\text{Width}} \\ \text{Pixel}_{21} & \dots & \text{Pixel}_{2\text{Width}} \\ \dots & \dots & \dots \\ \text{Pixel}_{\text{Height}1} & \dots & \text{Pixel}_{\text{Height}\text{Width}} \end{bmatrix}$$

Node

A Node data structure in this program is defined as:

```
1  typedef struct NODE_T {
2    char * name;
3    pixel color;
4    int fx;
5    int fy;
6    int radius;
7  } * node;
```

This node data structure is representative of a circular figure that will be drawn on the canvas before it is printed to the output file.

These attributes have the following meanings:

- ∇ `name` - The name of the node (i.e. what is printed centered in the circular node). Via libFreeType we are able to use any font for printing this name.
- ∇ `color` - A pixel representative of the color that the node will be drawn onto the canvas with.
- ∇ `fx` - The column value of the center of the circle (name coming from the focal point of a circle which is the center).
- ∇ `fy` - The row value of the center of the circle (name coming from the focal point of a circle which is the center). It should be noted that because of the definition of the canvas, a greater value here will place the circle lower in the canvas.
- ∇ `radius` - The radius of the circle representative of the node.

Drawing Nodes and Edges Efficiently

For this task I chose Bresenham's Circle drawing algorithm for its runtime in the class of $O(1)$ algorithms. Additionally I chose Bresenham's Line drawing algorithm for drawing the edges with a time complexity of $O(dx)$ where dx is the difference between the x coordinates of the endpoints of the line being drawn.

Drawing The Node Collection Legibly

One such idea that crossed my mind for this purpose is to draw the nodes circularly such that the end user can see the edges from one node to another and because in the Erdős-Rényimodel of random graphs relative position does not matter.

To accomplish this we must first find the point about which this larger circle will be centered (For this program's purpose:

$$fx_{\text{larger circle}} = \frac{\text{Width}_{\text{canvas}}}{2}$$

$$fy_{\text{larger circle}} = \frac{\text{Height}_{\text{canvas}}}{2}$$

Additionally, we must find the radius of this larger circle:

$$\text{Radius}_{\text{larger circle}} = \frac{\text{Height}_{\text{canvas}}}{2}$$

Next we find the angle between each node:

$$\theta = \frac{2 \cdot \pi}{\text{qty nodes}}$$

Finally to find the central point of each node to be drawn (assuming that they are linearly indexed):

$$fx_{\text{node}} = fx_{\text{larger circle}} + \text{Radius}_{\text{larger circle}} \cdot \cos\left(\theta \cdot \text{node index}\right)$$

$$fy_{\text{node}} = fx_{\text{larger circle}} + \text{Radius}_{\text{larger circle}} \cdot \sin\left(\theta \cdot \text{node index}\right)$$

Finally we draw each node.

Outputing Images

Using these data structures my program can output one of two file formats:

- ∇ `NetPBM` (Specifically `.ppm` files).
- ∇ PNG.