cg-classic-algos

Implementation of some classic Computer Graphic Algorithims.

Introduction

This is an implementation developed for the Computer Graphics class lectured by Rosilane Ribeiro da Mota in 2024/1. Using Tkinter for showing the pixel grid and implements the following procedures in python:

- Basic Transformations:
  - Translation;
  - Rotation;
  - Scale;
  - Reflection;
- Line Drawing:
  - DDA;
  - Bresenham;
- Circle Drawing:
  - Bresenham;
- Crop:
  - Cohen-Sutherland
  - Liang-Barsky

Author:

- Leonardo Caetano Gomide

The code is also available in the following repository: https://github.com/GomideLeo/cg-classic-algos

Implementation

The code is split in 2 main modules, `gui` and `shapes`, `gui` contains the main classes used for the GUI and `shapes` contains the classes that define each of the implemented shape, Point, Line and Circle. Also there is the `paintApp.py` file that defines the Tkinter app and its operations.

GUI

In the GUI module, we have 2 main classes, Pixel, that defines each pixel in the Grid, and Grid, that contains the pixels and defines the function for drawing the pixel grid on the Tkinter canvas

Shapes

In the GUI module, the shapes Point, Line and Circle are defined. Each of the shapes defines and is responsible the function for plotting itself, each of the 2d transforms and the crop function.

The shapes are defined by their mathematical descriptions:

```
Point:  position -> tuple(int, int)

Line:   start position -> tuple(int, int)
        end position -> tuple(int, int)

Circle: center -> tuple(int, int)
        radius -> int
```

2d Transforms

Each of the 2d transforms functions has the responsibility of taking the input parameters of the transform, and transforming the shape object inplace, so that the paintApp class is able to redraw them. The transformations that are not translation also have an origin so that if needed the origin point of the transform is changed

```
Translation:    x -> int
                y -> int

Rotation:       angle -> int (in degrees)
                origin -> tuple(int, int)

Scale:          x -> int
                y -> int
                origin -> tuple(int, int)

Reflection:     reflect_x -> bool
                reflect_y -> bool
                reflect_origin -> tuple(int, int)
```

For the circle scaling, the only action taken is to apply the transform to its center, the radius is not affected.

Plotting

For the plotting functions, each of the shapes recieves the grid and the canvas and is responsible for getting the pixel using the `grid.get_pixel` function and then calling the `pixel.set_pixel` procedure to set the pixel value correctly.

The exception is for the Line, that also recieves the algorithim its supposed to use, with `'dda'` and `'bresenham'` being the only valid values.

Cropping

For the cropping functions, each of the shapes recieves the `xy_min` and `xy_max` of the window and returns the shape that is going to be drawn by the paintApp in case something needs to be drawn, or `None` if the shape is outside the draw area

The exception is for the Line, that also recieves the algorithim its supposed to use, with `'cohen-sutherland'` and `'liang-barsky'` being the only valid values.

For the circle, only if the center is inside the draw area it will be drawn.

Running the Code

The code was created and tested on pyhton 3.10.13 using anaconda as a package manager, therefore is suggested to run in the same way.

First create and activate a conda environment with:

```
$ conda create --name <env> --file requirements python=3.10.13
$ conda activate <env>
```

or install on your python installation:

```
$ pip install -r requirements-pip.txt
```

After installing the dependencies, simply run the file `main.py`:

```
$ python main.py
```

References

- Donald Hearn and M. Pauline Baker. 1996. Computer graphics (2nd ed.): C version. Prentice-Hall, Inc., USA.