Project Description

SVG Reader

by Svetlin Popivanov

1st year, Software Engineering

Table of contents

Background

This is my first ever project. It was built after I finished my first year at university.

Overview

This program is designed to read svg files, perform actions on the figures inside based on the user's choice and then save those changes in the opened file (note that the changes may be saved in a different file or discarded). For simplicity, I have decided to pick the most basic shapes (rectangle, circle, line), but support for other shapes can be added as well. The coordinate system in this program is the following one:

- the positive x-semiaxis points to the right
- the positive y-semiaxis points down

The program supports the following operations:

- print – prints all the figures on the screen
- create – creates a new figure
- erase – erases an already existing figure
- translate – translates either one figure or all figures (translate all figures by default if the user hasn't specified)
- within – prints all the figures that are inside either a rectangle or a circle (depending on the user's choice) on the screen
- open – opens a file, reads its content, stores the information about the figures and the closes it
- close – closes the currently opened file and then the user cannot use any other command except for 'open' (note that this command is just checking that the file is closed – see command 'open')
- save – saves the changes to the currently opened file
- saveas – saves the changes to another file (it can be either new one or existing one)
- exit – exits the program

Description

Here, we shall present the user how he can interact with the program supported with examples:

1. After the initial start, the user can use only the command 'open' to access a file. He will not be allowed to use the other commands until he has provided the program with some information (figures).

Example:
open someFileName.svg
open C:\Temp\fileName.svg

When the user has provided a svg file (if it exists), the program opens it and stores the information about the figures and closes the file afterwards (so that we do not keep it open during the whole interaction between the program and the user).

Let's say that the file contained the following figures:

1. rect 75 80 10 10 green
2. circle 175 180 10 blue
3. rect 270 235 10 10 red
4. line 50 40 60 50 brown 5
5. line 30 50 30 40 yellow
6. rect 20 30 10 10 red

The information about the figures is as follows:

- rect (rectangle): x coordinate, y coordinate, width, height and fill color (optional: stroke color (word) and stroke-width (number) in this order)
- circle: cx coordinate (the x coordinate of the center), cy coordinate (the y coordinate of the center), radius and fill color (optional: stroke color and stroke-width in this order)
- line: x1 coordinate, y1 coordinate (the beginning point), x2 coordinate, y2 coordinate (the ending point) and fill color (optional: stroke color and stroke-width in this order)

2. After the program has closed the file, it unlocks the commands pointed out above for the user. They have the following functions:

➢ ***create:***

The user can create a rectangle, circle or line:

2.1. When creating a rectangle, the program needs an x value, y value, width value, height value and a fill color. The user can choose to provide the program with an optional information (stroke color or stroke-width, or both).

Example (correct):

create rectangle 20 30 70 70 red

Example (incorrect):
create rectangle 20 20 30

This will create the following rectangle: 8. rect 20 30 70 70 red

2.2. When creating a circle, the program needs a cx and cy values, a value for the radius and a fill color (the user can provide optional information as well)

Example (correct):
create circle 100 100 5 grey

Example (incorrect):
create circle 100 100 7

This will create the following circle: 9. circle 100 100 5 grey

2.3. When creating a line, the program needs an x and y value for each of the two connecting points (x1, y1 and x2, y2) as well as a stroke color (the user can provide optional information as well)

Example (correct):
create line 20 50 30 70 green

Example (incorrect):
create line 60 40 78 65 76 blue

This will create the following line: 10. Line 20 50 30 70 green

- ***erase***

When erasing, the user only needs to write the number of the figure that he wants to be deleted. The first figure is numbered 1.

Example (correct):
erase 2

Example (incorrect):
erase 10 (when the total number of figures is 6)
erase nine (digit should be provided not word)

- ***print***

  The program prints the figures like the above way (see command ‘open’). Note that this is a one word command so any other character/word/number after (or before) it.

  Example (correct):
  print

  Example (incorrect):
  print 2

- ***translate***

  Using this command, the user can shift the position of a figure or all figures. The user can move the figures horizontally and/or vertically. The program needs at least one value (either horizontal or vertical translation value or both). Both of these values can be positive or negative.

  Example (correct):
  translate vertical=10 horizontal=10
  translate vertical = "10"
  translate vertical = -10 horizontal = " -10 "

  Example (incorrect):
  translate vertical = 14fy5 horizontal = 15
  translate vertical = ' 10 '
  translate vertical = 20 horizontal = " - 20 "

  <u>Note</u>*:* The user can specify vertical before horizontal as well.

- ***within***

  The program takes a figure and checks if the figures from the file (and those that have been created by the user) are inside it. The only two figures the user can specify here are rectangle and circle. When checking with rectangle, the program will only need x coordinate, y coordinate, width and height values. When checking with circle, the program will only need the cx coordinate, cy coordinate and r value.

  Example (correct):
  within circle 0 0 5
  within rectangle 20 20 20 20

Example (incorrect):
within rectangle 20 20 20 20 yellow
within circle 0 0 5 green red 6

- ***close***

This program closes the currently opened file (if there is any) and erases all used information up until that moment. Afterwards, the user cannot use any other command than 'open'.

<u>Note</u>: this is a one word command

Example (correct):
close

Example (incorrect):
close now

- ***save and saveas***

When the user uses either one of those commands, the program saves the changes (save: in the currently opened file; saveas: in another file).

<u>Note</u>: The user has to specify a file name when using saveas and save is a one word command.

Example (correct):
save
saveas fileName.svg
saveas C:\Temp\fileName.svg

Example (incorrect):
save now
saveas filename

- ***exit***

When the user uses this command, the program stops, all information is cleared and the user must press any key in order to close the window.

Note: this is a one word command

Example (correct):

```
exit
```

Example (incorrect):

```
exit now
```

Software Used

Microsoft Visual Studio - https://visualstudio.microsoft.com/