Simple Drawing App

Matias Tarmo 868925
Computer Science
1st year
18.5.2022

Outline

I have made a simple drawing app with the features that are mentioned in the intermediate drawing app description. You can create simple drawings with the app. You can choose from drawing a line, circle, ellipse or rectangle. The color of the shape can also be chosen from many options on the RGB spectrum. While drawing the shape will preview itself when you are holding the mouse and after you release it, it will show onto the canvas. After drawing you can also revert the changes that you have made. Saving and loading is also possible. It happens with xml files.

Manual

The program is easy to use. Just run it from the main file and everything else happens from the UI. You can choose what shape you want to draw. The default is a line. Under that you can choose the color for your shape. Under the color we have the undo and clear buttons. They do exactly what they say they do. On top we have the file menu. From that you can save, open and make new files. Saving and Opening open file menus where you can name and choose the file. (At Least on mac you have to add .xml to the files so that they save to xml files and can be opened)

Structure of the program

I have structured the program in three main parts. We have the main object which does most of the work. Then we have the trait and object Drawable on which we build the different shape classes. Every shape has the draw method which draws the shapes on the canvas. All shapes other than clearedBoard also have values for the coordinates on which they will be drawn on. So when on the main object we create the objects we give them the coordinates, color and the graphicContects. Cleared board only takes the list of current drawables and the graphiContects. This is so we can undo our cleared board. All of these shapes also have a toXml method that converts the data to xml so it can be saved to xml. xml sounded like the best for handling text based files, because it's easy to edit in scala.

On the main object I do all of the rest. We choose the color with the built in colorpicker and store the value in a var. The chosen drawing type is chosen with buttons and then the mouse events are executed accordingly on the chosen value. This is done with if and if-else. It could

have been done in match-case but in this case if-else was simpler to do. Undo and clear are simple onAction events that happen when the button gets pressed.

Saving and loading is done with xml. When saving we use the shapes' toXml method to make them to xml and then we save them with the file chooser to the place where the user wants to save it. Color is saved with the xmlToColor method. Loading loads with fileSelector. It has to be xml and correctly formatted to work.

The UI is very simple. It's just a scene with three borderpanes. Top being the file menu. Left side is the drawing selection and the middle is the drawing canvas. It could have been done with split panes so it would scale better.

My old UML is mostly correct on my finished program. It would only be simpler because I don't need the pixel class. We also don't need classes for the things that would have been done in the harder version. Also the file saving and loading was made in the main object. Otherwise the UML would be currently the same.

Algorithms

The main algorithms are when the shapes are being drawn. First we have the start coordinate and the end coordinate. Then we calculate how the shape should be drawn with builtin methods. Loading and saving also have a simple algorithm that saves or loads the correct shape based on their type.

Data structure

I used Lists' to save data. That is because they are sorted so implementing the undo was easy. Also when getting data from them it was nice to have it sorted. List are also efficient enough for my use case

Files

I saved my files on XML. XML are readable for humans and they are easy to handle in scala. Few examples can be found from the same folder as this file.

Testing

I did all my testing with the graphical interface. I didn't write any unit tests. They could have been useful in loading and saving files. The basic drawing functions were easy to test with

the UI because you can see if something is wrong with them and fix it. The code works with all the basic use cases that I tested manually.

known deficiencies and faults.

The biggest problem currently is with saving and loading. They only handle correctly formatted files. Another one is that the app is not scalable at all. The canvas is fixed in size and it is hard/impossible to distinguish where it ends or starts. The program also randomly crashes sometimes. I have yet to find a cause for that. The code also contains fragments that are repeated multiple times. I would have liked to make some method for them, but couldn't get them to work.

Best and worst parts

I like the conversion of color to xml really much. I think it's an elegant solution to problem that I thought would be hard because Color couldn't be made to string simply.

I also like the way I draw the circle. That is because for the other shapes I use predefined methods. For the circle I calculate the position with math.

Bad part is the loading and saving. The code is repetitive and it handles no errors. I don't have much experience with xml so I'm not sure how I would have fixed it.

Scheduling

My schedule didn't meet any of my own targets. I was late already as I started about one and a half months ago. It took me a long time to get going with scalaFX. I used a few days for that. After that it took me about a few hours a day for a week to get the drawing functionally done. After that I dove into XML. That also took longer than expected. I struggled with error handling for a few days and after that I decided to ignore errors. So all in all XML took me about a week to get done. I was sick close to the deadline and couldn't get the savings functionally to work and I didn't have the energy to write the documentation so the return ended up really late. The order I completed the tasks followed my plan.

The final evaluation of the project

All in all I liked my project. It fulfills all the main requirements that were set for it. Good things are all the basic functionalities are coded simply and they work well. The canvas could be resizable, so that the app would look nicer. The Canvas should also be more distinct so you can see where it ends. The xml conversion of colors and shapes is well executed in my

opinion. Saving and opening the file is also not so great, due to no error handling. In the future I would like to solve the problems I have mentioned here.

The data structures are good for the use case and I wouldn't change them. XML files are also great for saving to text. Classes are also good with the Drawable trait so they can be used in the same collections. I think my current program is easy to expand. Everything is made simply and it's easy to understand.

If I started the project now the only thing I would change would be that I'd start earlier.

References

(https://www.scalafx.org/docs/home/)
(https://github.com/scalafx/ProScalaFX/tree/master/src/proscalafx)
(https://github.com/scala/scala-xml)