Project Document - Saku Suorsa

Personal information:

Title: Drawing Program

Student's name: Saku Suorsa

Student number: 1011564

Degree: Computer Science (B.Sc. + M.Sc.)

Year of studies: First year

Date: 26.4.2022

General description:

My project is a drawing program. The program works similarly to other simple drawing programs. The user has an option to choose the color and size of their brush. The user is be able to draw the following shapes: line, circle, rectangle and oval. The drawing happens by dragging the mouse and the shape to be drawn will show already in the user's screen as the user is drawing.

The program will allow users to use basic tools such as undoing the last change made and clear the entire drawing or delete selected objects. Objects can be grouped as well to make them a "one bigger object". The chosen elements can be modified, moved as well and deleted as well.

The user can open multiple drawings simultaneously and when making changes to one drawing, others are not affected. The drawings can be saved to a text file named by the user. The program can open the file containing the drawing data so the user can continue their drawing later.

The program will also allow users to write text on top of the drawing with a configurable size. I may also add some more advanced drawing tools like a perspective tool.

The user can use a couple of simple commands on the console to help the drawing process.

The main changes for the plan was that I added a tree structure of the drawing and a box to display selected item's properties to the right in the GUI. I also did not do some classes or did some class design otherwise mainly because I did not know exactly what can be done easily with scalafx libraries when I made the plan.

The work was completed on the demanding difficulty.

User interface:

The program automatically generates a graphical user interface when the program is run. One empty drawing gets added automatically and the program is automatically on draw mode so the user can start drawing immediately when the program is opened.

On the very top bar of the user interfaces there are two menus. File menu and Edit menu. From the File menu the user can create a new drawing, save the current drawing, open an existing drawing and quit the program. From the Edit menu the user can undo the previous change and redo the previous undo.

On the left, there is a gray sidebox. It contains the tools needed to draw. On the top of the sidebox there is a menu button where the user can choose the mode in which they want to use the program. The possible modes are Draw, Edit, Text and Select.

When the program is in Draw mode, the user can draw by holding the mouse button down on the canvas that is in the middle of the GUI. The shape and its properties get determined by what has been selected from the tool selections on the left box.

The first drawing tool important in drawing is the color selector. It is found right below the mode selector. The selected color determines the color of the drawn object.

Below the color selector is a shape selector. This determines the shape to be drawn. The options available are line, rectangle, ellipse and circle.

Below that there is a checkbox that determines whether the drawn shape is filled with paint or not. This does not apply to lines of course but it does on rectangles, ellipses and circles.

Below the checkbox there is a brush size selector where the user can enter the brush size they want to use when drawing.

Below that there is a similar field but for font size. This determines the font size of the written text.

On the bottom there is an add group button which adds a group to the currently chosen group. In my program groups can be added inside of groups and the group methods work recursively so they are applied to every descendant of said group.

When the program is in Edit mode, drawing cannot be done. In edit mode the user can choose one object on the program by clicking it. The selection is highlighted by a light blue outline. The chosen element gets selected from the tree view on the right as well. The selecting can be done directly from the treeview on the right (even in draw/text modes).

The user can also select an entire group of objects. This happens by clicking the desired group on the tree view. When a group is selected the objects of the said group are highlighted. An exception to this is the root group. Root group is often selected as default so I made a design choice not to highlight its objects because otherwise the entire drawing would be highlighted very often.

When an element or group has been selected it’s properties open on the bottom right box below the tree view. There the currently chosen object or group can be edited. When in edit mode, the selected object or group can be moved by dragging the mouse on canvas.

Text mode is simply for writing text. When in text mode, the user can add text to a drawing by clicking on the position they want to add text on. Then a pop up window is opened where the text to be added is written to. The color and font size are decided from the tool set on the left box similarly to drawing.

Select mode is similar to edit mode. However, in select mode the objects can’t be selected from the tree view at all but now multiple objects can be selected by clicking on them and unselected by clicking on them again. Now on the bottom right below the tree structure is shown how many items are selected and their properties can be edited. Selected objects can be moved around by dragging the mouse through canvas similarly to edit mode.

In the bottom center is a black box which is a console. The user can enter there a couple of commands. The commands available can be seen by typing help to the console.

User can adjust the windows in the GUI as they wish.

Pictures of the user interface can be found from the appendices section.

Program structure:

The main parts the program is split to is the user interface, drawing and its contents, undo redo management, drawing and being able to edit the drawn objects, mouse input, button actions, file management and commands.

The final realized class structure is that I have classes Main, Drawing, UndoRedoManager, Command, Constants, Style and then I have traits Drawable and UndoRedo and then I have a lot of classes for both of them that implement said trait.

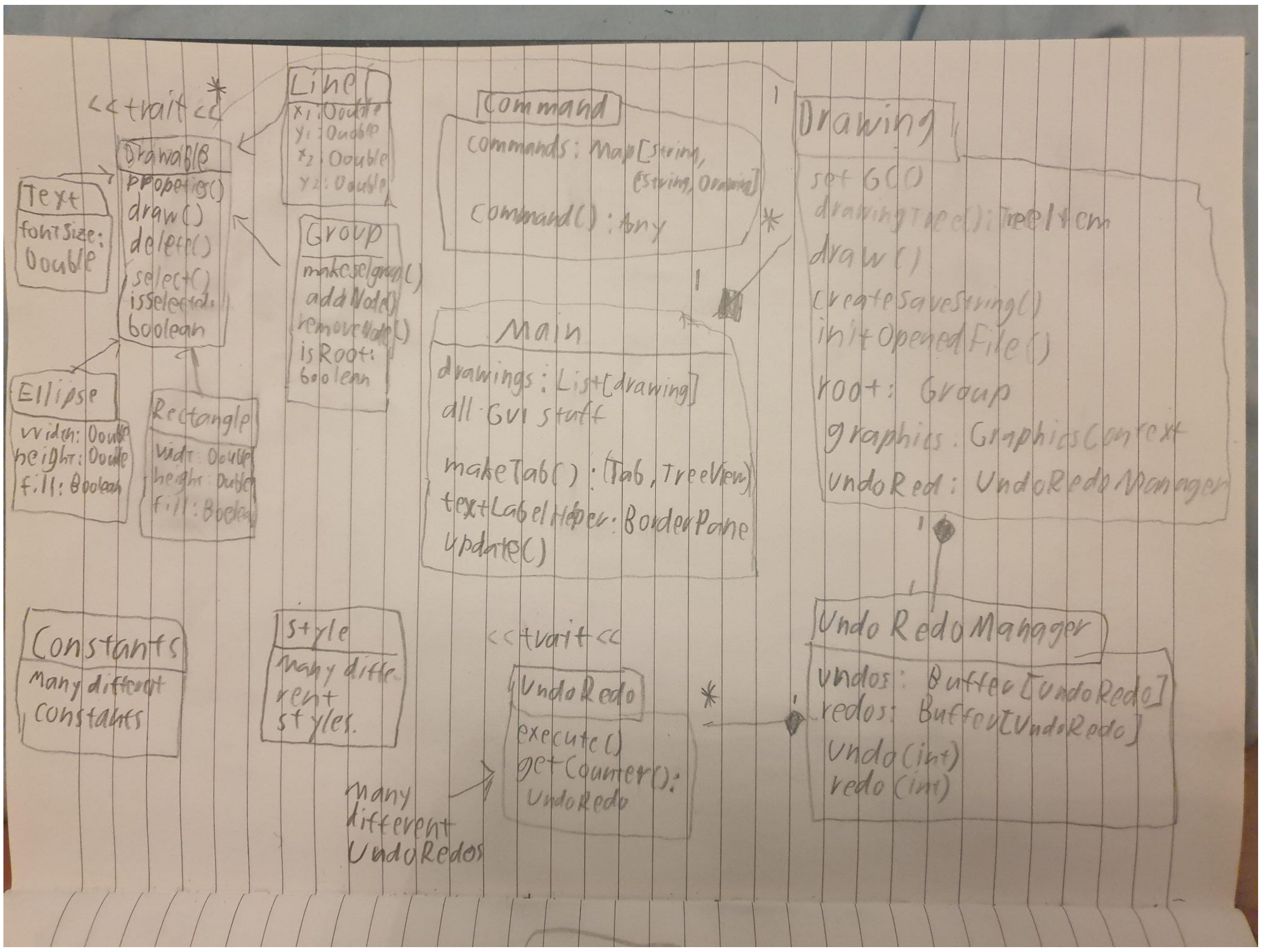
In my main class I construct the user interface, handle the mouse input and button actions. Main class uses the Command object to handle commands given to the console. I realize that a problem with this structure is that I have too much stuff in Main but unfortunately I did not have time to refactor that.

Main’s user interface can have multiple drawings (multiple drawings can be simultaneously open). Drawing can contain Drawables in a tree structure to describe the drawn components. Drawing and Drawables take care of drawing and being able to edit the drawn objects afterwards. Drawing also takes care of opening a drawing from a file.

Each Drawing has its own UndoRedoManager. UndoRedoManager can contain UndoRedos (undos and redos).

Almost all classes use information stored in Constants and Style.

A UML chart of my design below. The chart includes the relations and most important attributes and methods of the classes.



The key methods introduced: Mains makeTab: (Tab, TreeView[Drawable]) method which creates drawing and a TreeView associated with it. Drawing has createSaveString(): String which calls its drawables similar methods to create a save string that is written to the file. Drawings initOpenedFile() method initializes a new drawing from the lines of the file. Drawables draw() method draws itself to its drawing. UndoRedo:s execute() method executes the given undo/redo and its getCounter() method gives a new UndoRedo that counters it.

Algorithms:

In this program I used a lot of recursive algorithms to manage my tree structure. I had to develop algorithms to add an item to a wanted place in the drawing, to select a wanted item in the tree and to remove a wanted item from the tree for example.

When drawing with my program I need to add the drawn item to the group that is selected in the tree or in the parent group of the selected item if the selected item is something else than a group. I then check if the selected is a group, if it is, I add the new drawn object to its children. If it's something other than a group, I add the drawable to the selected drawable's parent's children.

When selecting a wanted item, I go through every child of every group in the tree and check if it matches the item I want to select and then select the said TreeItem from the TreeView. This may be a bit efficient but I could not figure out a more efficient way to do this with Scalafx TreeView.

When removing, I first select the item with the selection algorithm and then remove it from the selected parent's children list.

My undo redo algorithm is made so that whenever I make a change, I add a new UR item to the drawings UndoRedoManagers undo list. The said UR item when executed cancels the change and creates its counter and puts it into the redo list. This, when executed, cancels the said undo and puts it back to the upcoming undolist and so on.

I implemented algorithms for drawables that check if a drawable contains a mouse clicks coordinates so it can get selected. I used some basic math to calculate this for each drawable. I had their coordinates, width, length and so on known so it was not that difficult to do. I added some room of error for clicks for lines with small line width for example.

The drawing algorithm itself was quite basic as well. The starting and ending point of a mouse drag are known so some basic math can be used to calculate the coordinates, widths and heights of each items.

Data structures:

The main data structure used in this program is a tree structure consisting of drawables. Every drawing opened with the program has its own tree structure. Every tree structure's root is a group item. Only groups can have children in the tree.

I used the tree structure to make it possible that groups can have groups inside of them and the drawn objects can be edited as individuals if wanted even if they are in groups.

In my program I use mainly immutable lists to store children of menu items or groups for example. In menu items immutable lists are fine because you do not have to usually add anything to them when the program is run by the user. In undo and redo collections I use mutable buffers. They are useful because their containments get constantly changed.

In this program it was not that important which data structures I use for storing the children of the groups and undo/redos because efficiency is not a key question because in real use cases not that much stuff is stored into these. I used the data structure that I felt like I was comfortable with when solving the problem I had. I could have very well used immutable lists for undos and redos and mutable lists for menu items. It would not have made too much of a difference.

However, I think the tree structure for storing the objects in the drawing was necessary if I wanted the program to work the way it works. There would have been others and probably a lot easier ways to store drawables. It could have been done with just a one list but then you could not have put groups inside of groups and show the tree structure in the right of GUI and select the wanted items from there like is done in my program.

For some specific undos I use maps. For example when I have to be able to undo a change I made to the multiple different items (groups or select mode), I have to map each drawables individually to their original values so I can return the original state of each of them when undo is called.

I use map for commands as well. That way I can easily check if the command exists by iterating through the keyset and all I have to do to add a command is to add it to the map.

Files and Internet access:

The program allows the user to save the drawing to a text file and open the text file containing the drawing data by the program.

The file is formatted as follows. The first line must be equal to a key string *"l0rkcyAaSm0buQ9OBov5"*. If it is not, the opening process is not continued. This is done to prevent the user from accidentally opening other text files that are not created by this program and are wrongly formatted. This can prevent an exception that would occur from trying to handle an incorrectly formatted text file.

Every line after that contains an object. The different objects are formatted as follows:

group,n

This is a formatting for a group. It tells that an object to be added to the drawing is a group and n tells how many descendants it has(how many lines in the file are objects under it). Note that every drawing has a root group which contains everything in the drawing.

rectangle,x,y,width,height,colorstring,linewidth,fillstring

Formatting for a rectangle. Everything after the rectangle are strings of doubles, other than fillstring which is a string of either 1 or 0 and colorstring which is a string formed as follows: red_green_blue_opacity.

ellipse,x,y,width,height,colorstring,linewidth,fillstring

This is a saving format for ellipses and circles. Note that the circle is a special case of ellipse so it can be saved as an ellipse. Works similarly to the rectangle.

line,x1,y1,x2,y2,colorstring,width

Formatting for lines. Everything after line is a string of double except colorstring which is similar to colorstring explained in the rectangle part.

text,x,y,text,fontsize,colorstring

Formatting for text. Everything after text is a string of double except text is just a string and colorstring formatted similarly to other colorstrings.

An example drawing file could be as follows:

```
l0rkcyAaSm0buQ9OBov5
group,11
line,144.59999999999997,187.0,205.39999999999998,287.0,0.0_0.0_0.0_1.0,2.0
group,7
rectangle,347.0,215.8,88.80000000000001,56.80000000000001,1.0_0.0_0.0_1.0,2.0,0
rectangle,496.59999999999997,157.4,77.60000000000008,50.400000000000006,1.0_0.0_0.0_1.0,2.0,0
group,4
ellipse,747.0,325.4,76.00000000000003,76.00000000000003,0.0_1.0_0.0_1.0,2.0,1
ellipse,378.2,338.2,80.79999999999995,80.79999999999995,0.0_1.0_0.0_1.0,5.0,0
group,1
text,612.5999999999999,246.19999999999996,TestText,25.0,1.0_0.7019608020782471_0.7019608020782471_1.0
text,173.39999999999998,91.0,Testing,20.0,0.0_0.0_1.0_1.0
ellipse,371.8,95.0,93.60000000000002,30.400000000000006,0.501960813999176_0.0_0.0_1.0,5.0,0
```

This drawing program does not use the internet at all but works completely offline.

Testing:

The program was tested by trying the things out in the user interface after they had been implemented and when I thought the features were ready a bit more heavier testing was done. This was done by me trying to use the program doing my best to purposely try to break it.

The drawings use a tree structure where the components and groups are added. Groups can contain groups so my program has to use a lot of recursive methods. I realize these can be inefficient so I tried to create big drawings and test is there any inefficiency notable to the user. I could not find any inefficiency that was really noticeable by eye when doing editing, selecting, removing and other methods on as big drawings as I had energy to create by hand.

The program passed the tests except for some extra features which I tell more about in the next known bugs and missing features section.

I really would have liked to generate a very large drawing file by computer to test how the program can handle very large drawings but unfortunately I did not have time for that.

I did not do any unit tests because I did not really see any place in my program where they would be necessary and I did not really have time to do much extra.

The testing was done quite similarly to what was planned except I did not do unit testing.

Known bugs and missing features:

The program is not really missing any features required from the demanding project description. It is missing some other nice to have features though. One of them is that when the user quits the program it would be nice if the program tells him to save the unsaved files so there would not be accidental loss of progress. Another feature I would have liked to implement is free draw but since it was not in the list of features the project had to have it was not prioritized and I ultimately did not have time to implement it. A third nice feature could have been to be able to break down groups so that the group gets deleted but the children of the group stay. Now there is only an option to delete the entire group.

One known bug is that when the user opens a new drawing, the start positions of the windows are not the same as they are in the first drawing. This is a very minor visual bug because the user can easily adjust the windows however they like anyways. I do not really have an idea what causes it though because I use the exact same makeTab function with the same parameters to create a new drawing than I used in the first one.

Another bigger bug is in the more complicated editing features (rotate, shear, translate and scale) I added to the groups. They seem to do what they should perfectly fine but after they are used, the coordinates of the objects are changed so it makes moving the objects really weird. I sort of know what causes this. I think it has to do with messing with the graphics context but there did not seem to be an easy fix. However, I wanted to leave these features in the program anyway because they can be fun to use and work well when adding a final touch to the objects and not moving them after. The objects can be moved normally after the rotate/scale/translate/shear values have been put back to zero. I added a red warning that these features cause bugs. I did not have time to add undo features on these even though it would have been quite simple because the features were not necessary and they were in beta testing and thus not prioritized. I did not add the information of rotate/scale/translate/shear to the files either so they don't get saved. It should be considered as more of a test feature to maybe look at your drawing from different angles while deciding what to draw next.

Another thing to note as well is that when editing a group, by adding a new value to the property field (egg. brush size) and pressing enter and then clicking away from it causes two actions and thus two undos have to be made to cancel that. A proper way to update the value field is to just click away from it and not press enter. I fixed this for singular objects that an undo is not created if the updated value is the same as the previous value but I could not do this for groups because the said previous values are not necessarily the same for all group members.

This is not necessarily a bug but a visual thing to note that the order of children in a tree structure can become different than they originally were after making some changes and undoing to the previous state. It does not matter since the order of the children of the parent node does not matter at all and every child is "equal". It can just be visually a bit weird if paid attention.

This is not necessarily a bug either but worth to note that the values seen in the groups's property panel (bottom right when group is selected) are not necessary the values of the group. They are just used to update groups items. Since group items can be edited as individuals as well it is impossible to define a groups color value for example since the colors can differ between the group's members if the user so wishes.

3 best sides and 3 weaknesses:

The 3 best sides of my program are the user interface, tree structure for drawing and undo redo system.

I think the user interface is one of the best aspects of my program because it is graphical, quite easy to use and it looks like a drawing program. The user can also modify the size of the windows as they prefer. New drawings can be added there in a nice way and the selecting and editing objects can be done quite nicely. It feels nice to select and unselect objects with the mouse and move them around. Clear indication is given on what has been selected and other editing can be done in a nice way as well.

Tree structure for drawings is one of the strong sides of this because it allows the groups to be used better than what was required in the requirements. It allows the user to add groups inside of groups and objects can be modified individually as well even if they are in groups. Group editing is also possible of course. The principle is that the latest change always overrides. Another reason I think the tree structure is one of the best sides is that its implementation required a lot of sweat and tears. Because groups can contain other groups it made the implementation of many very simple things like removing and changing color really difficult since you had to use recursion a lot. Not to mention selecting items across the groups in select mode and having to make saving/opening to a file and management of undos possible.

That's where we get to the third best side, undo redo systems. Undo redo system required quite a lot of work because every change had to be undone and there were a lot of different kinds of changes the user could do because of the aforementioned facts of having groups inside of groups.

The 3 weaknesses on the other hand are a way too big main class, not well enough refactored and commented code at some points and the advanced group editing methods rotate, translate, shear and scale.

The too big main class could be fixed by dividing it into other classes. Another class could be made for user interface, another one for the logic of buttons maybe and one for handling mouse input.

The code could be refactored more in the other parts as well. There are multiple add to and remove from methods implemented that do essentially the same thing. Add to or remove from the drawing tree.

I should have definitely commented on the code more, especially on those recursive methods. I myself understand very well what does what and what can be found where but some of the methods and functionality are probably quite difficult to understand for others trying to read to code.

If I had had one day more I would have made these things in a lot better shape but unfortunately the deadline arrived.

Third weakness, the more advanced editing methods are more difficult to fix. I do not have a solution for the coordinate mess up they cause as of now completely. I could do the scale and translate methods easily without using the graphics context but instead editing the properties of the objects of the group and that would work. I do not know what I would do with the shear and rotate though.

Deviations from the plan, realized process and schedule:

I did not follow the plan that much when it came to schedule. I had way too much stuff going on on other courses so I could not start my project during the first two week period. I had a lot of stuff going on the next two weeks as well so I could only work a couple of hours with the project during that time. During that time I only got started with the basic user interface. During the next two weeks I worked a lot. I got a lot of stuff done. The user interface was almost in its final shape and the basic mechanics of drawing and other classes were there. During the final two weeks I worked hardest. I did a lot of all nighters and put hours and hours into this project. During the last two weeks I added an undo redo system, command handling, file saving, select mode and fixed a lot of stuff I found broken from the last one. It is not that well defined what features I did in which period since with a lot of features I started to implement them in one period, worked on something else then and later came to finish or fix the features I started before.

The process derived from the initial plan mainly so that in my plan I had clear steps. For example, first I do this stuff then this stuff but in reality it was more like I started doing one thing, worked with another and then came back to the older thing to fix it / update it / finish it.

I underestimated the time the project would take a lot. My estimation was 37 hours but that time was almost spent before the last two week period in which I worked almost almost double the amount I had previously worked combined. So the total would have been more like 80 hours but it is quite hard to say exactly because a lot of times I just worked an all nighter in a flow state not looking too much to the clock just developing.

Biggest underestimations came from the fact that I did not know how difficult implementing some methods I thought were quite basic, selecting, removing etc because of my tree structure that could have groups inside of groups. Tracking some quite simple bugs took surprisingly long as well.

The main thing I learnt in a process is how easy it is to underestimate how long a programming process takes. It would be wise to reserve some more time than planned just in case.

Final evaluation:

Overall I am quite happy how this project turned out. I got implemented everything that was required for the demanding project without necessarily any shortcomings. There are several features I would have liked to implement that I did not have time for such as free draw, having the unsaved drawings saved when the program is closed, having the possibility to destroy groups while not deleting the children etc. I would have also liked to separate stuff from the main class better, refactor the overall code better and better comment on some complicated recursive methods but at some point the deadline comes and then you have to call the project ready.

I think the class structure is fine when it comes to drawings, drawables (different shapes and groups), different undos and undomanager, having constants in one class and styles in one but where the class structure should be bettered is that the main class should be splitted to at least a differing class for the gui, another class for handling mouse inputs and another handling the logic of the buttons etc. I had the main the classes separated better in my original plan but when I started to work on the project I started to construct the gui to the main class like was done in the scalafx tutorials and later it became harder to differentiate from it and I did not have time / courage to do it at the end of the project.

The structure is quite suitable for making certain additions. New menu items can be easily added to different menus. New commands can be easily added to the command object. New drawables can also be added although this requires a bit more coding since the drawable trait contains quite a lot so a lot of methods should be implemented and some work will go to defining the properties since the properties have to be editable from the user interface. That being said, adding a new property to an existing shape is quite easy.

If I would start over from the beginning, I would separate the gui and other stuff from the main class already at the very beginning.

References:

https://www.scalafx.org/

https://otfried.org/scala/gui.html

https://otfried.org/scala/gui.html

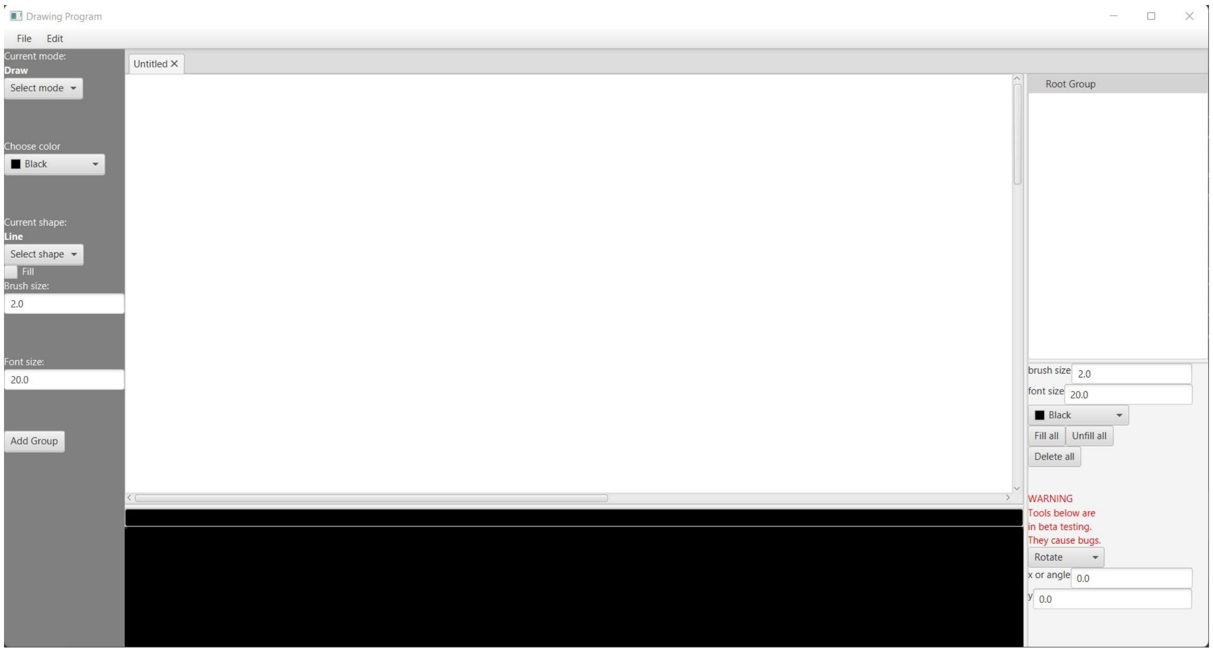
https://www.youtube.com/user/thenewboston/featured

https://www.youtube.com/c/MarkLewis/featured

https://www.youtube.com/c/BroCodez

Appendixes:

The GUI when the program has been started.



Current mode:
Draw

Choose color
Black

Current shape:
Line

Brush size:
2.0

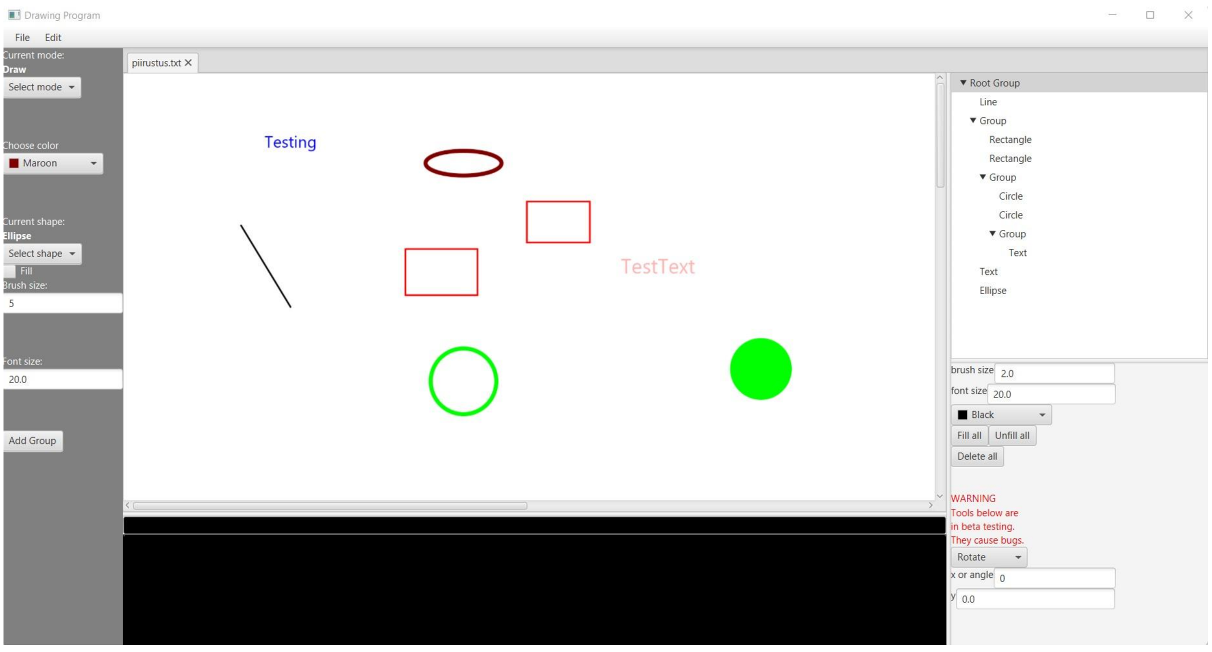
Font size:
20.0

brush size 2.0
font size 20.0
Black

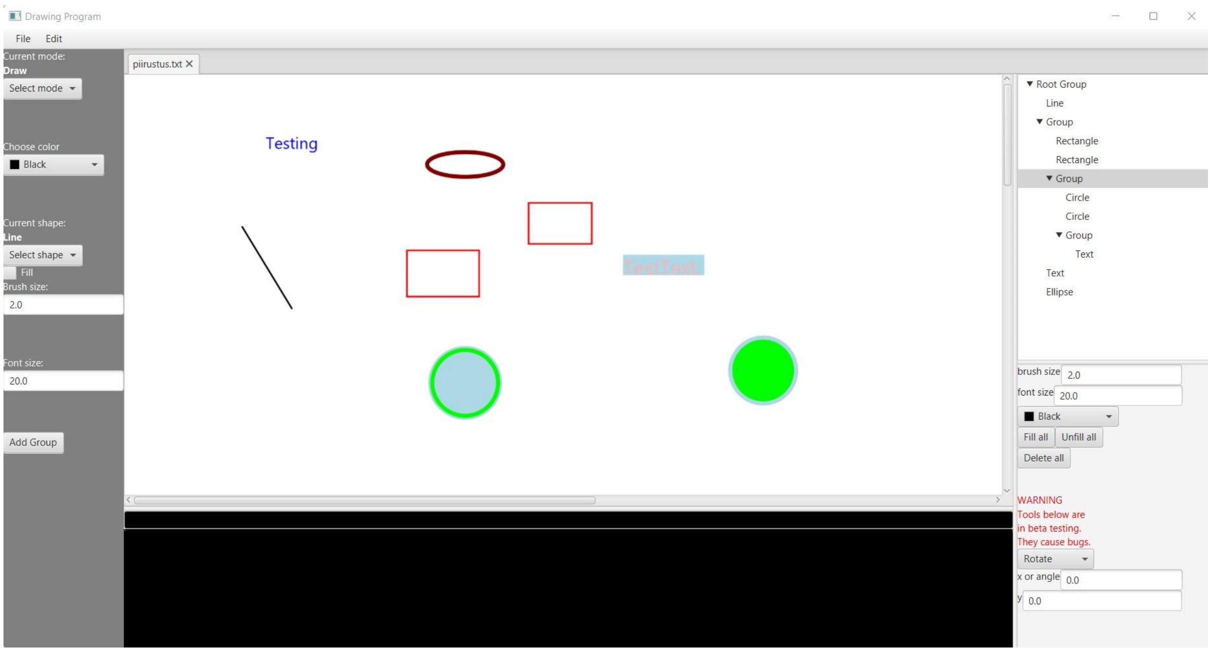
WARNING
Tools below are
in beta testing.
They cause bugs.

x or angle 0.0
y 0.0

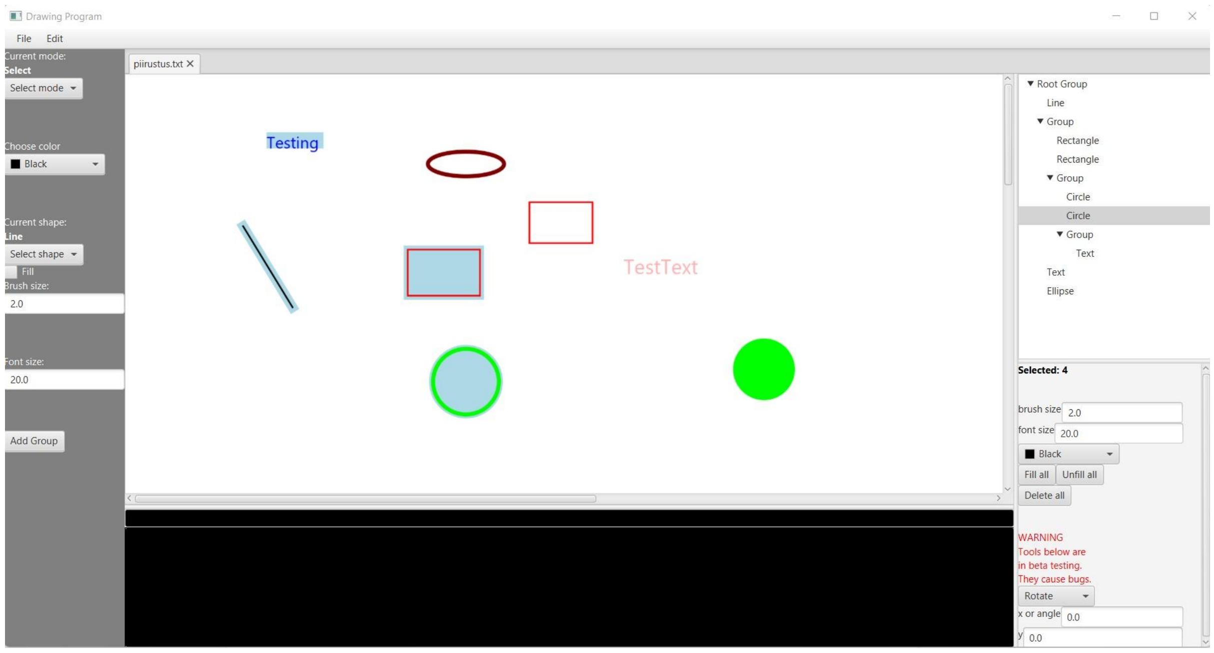
The GUI after some drawing has been done and groups and text added.



The members of a selected group are highlighted. Note that the selecting and upper group selects also the items that are in groups below it.



Objects selected across groups in Select mode.


Drawing Program

File Edit

Current mode:
Select
Select mode

Choose color
Black

Current shape:
Line
Select shape
Fill
Brush size:
2.0

Font size:
20.0

Add Group

piirustus.txt

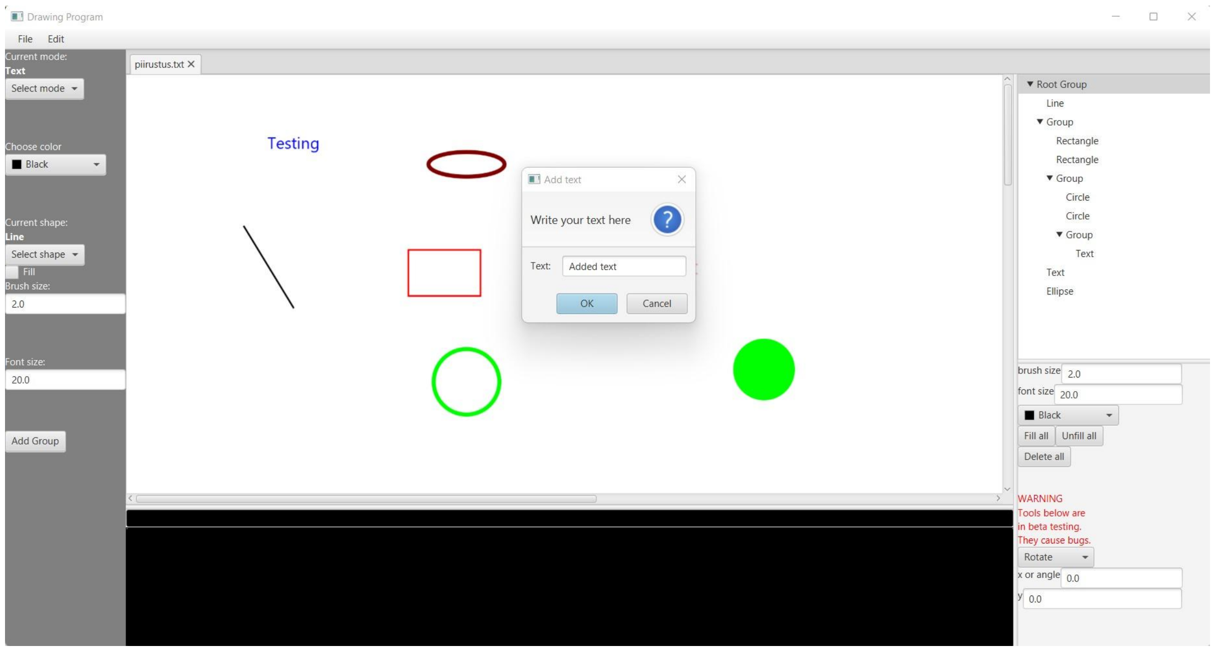
▼ Root Group
Line
▼ Group
Rectangle
Rectangle
▼ Group
Circle
Circle
▼ Group
Text
Text
Ellipse

Selected: 4

brush size 2.0
font size 20.0
Black
Fill all Unfill all
Delete all

WARNING
Tools below are
in beta testing.
They cause bugs.
Rotate
x or angle 0.0
y 0.0

Text being added.


Drawing Program

File Edit

Current mode:
Text
Select mode

Choose color
Black

Current shape:
Line
Select shape
Fill
Brush size:
2.0

Font size:
20.0

Add Group

piirustus.txt

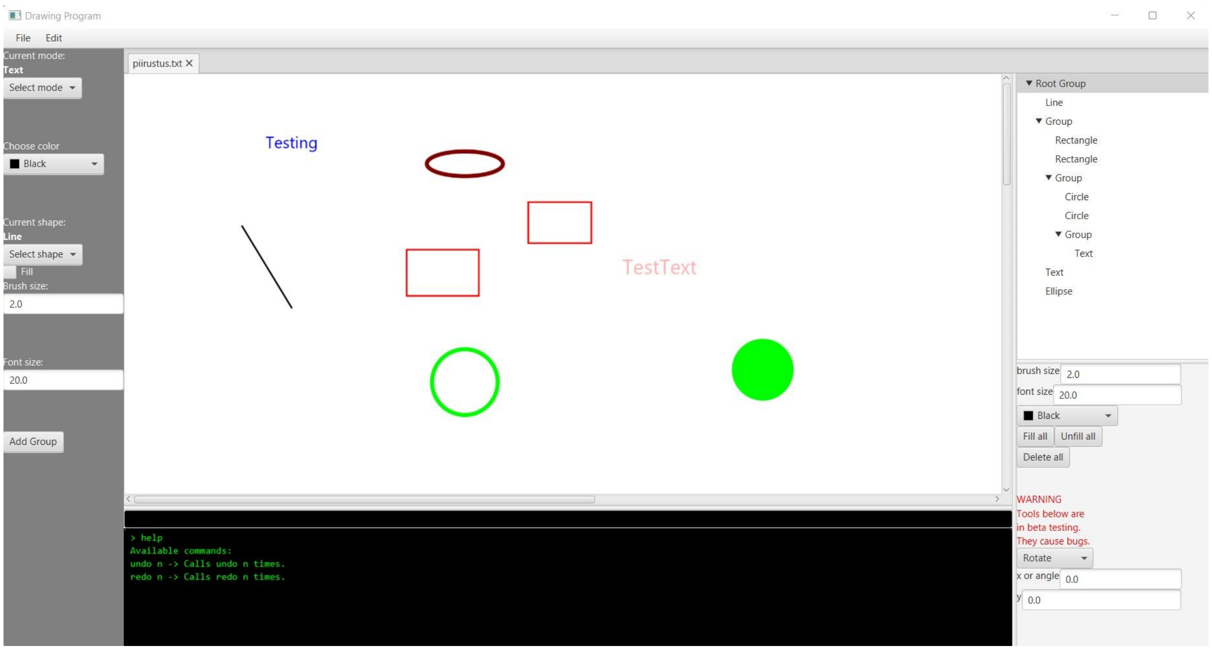
Add text
Write your text here
Text: Added text
OK Cancel

▼ Root Group
Line
▼ Group
Rectangle
Rectangle
▼ Group
Circle
Circle
▼ Group
Text
Text
Ellipse

brush size 2.0
font size 20.0
Black
Fill all Unfill all
Delete all

WARNING
Tools below are
in beta testing.
They cause bugs.
Rotate
x or angle 0.0
y 0.0

Help command entered to console.



Current mode:
Text

Choose color
Black

Current shape:
Line

Brush size:
2.0

Font size:
20.0

brush size 2.0
font size 20.0
Black

WARNING
Tools below are
in beta testing.
They cause bugs.

x or angle 0.0
y 0.0

```
> help
Available commands:
undo n -> Calls undo n times.
redo n -> Calls redo n times.
```