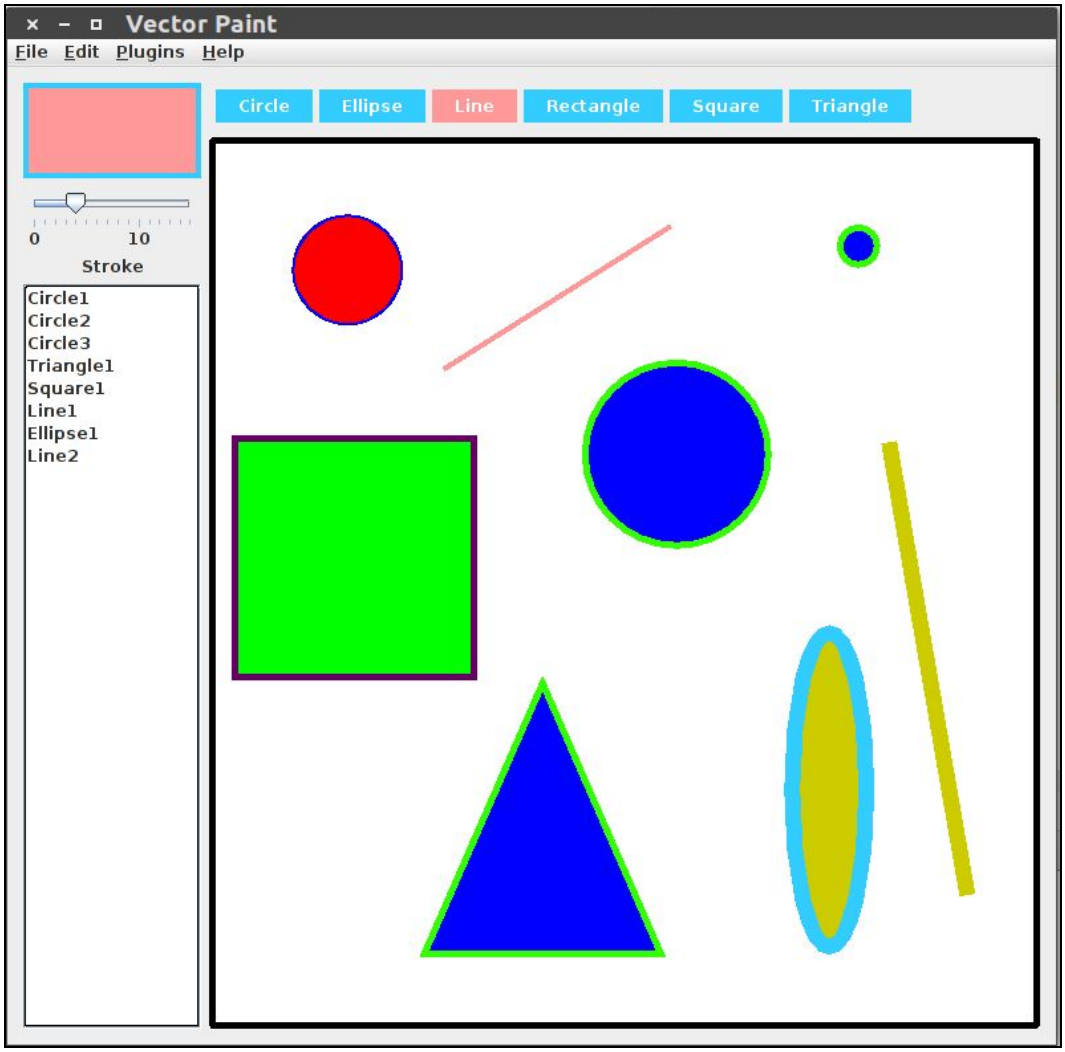
PAINT REPORT

Assignment 2: Vector-Based Drawing Application

2nd Year, CSED, Alexandria University



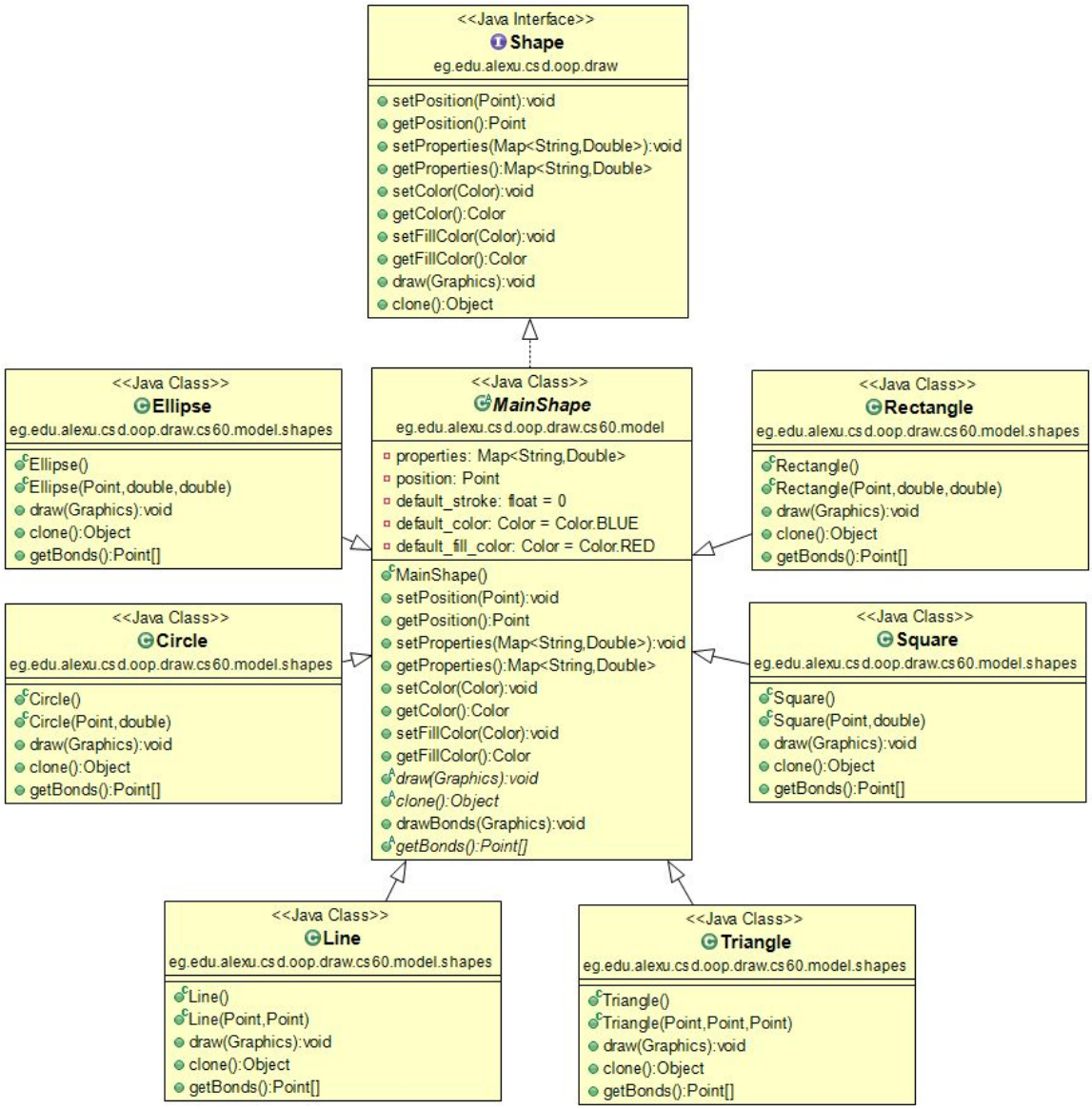
Mohamed Mashaal (60)

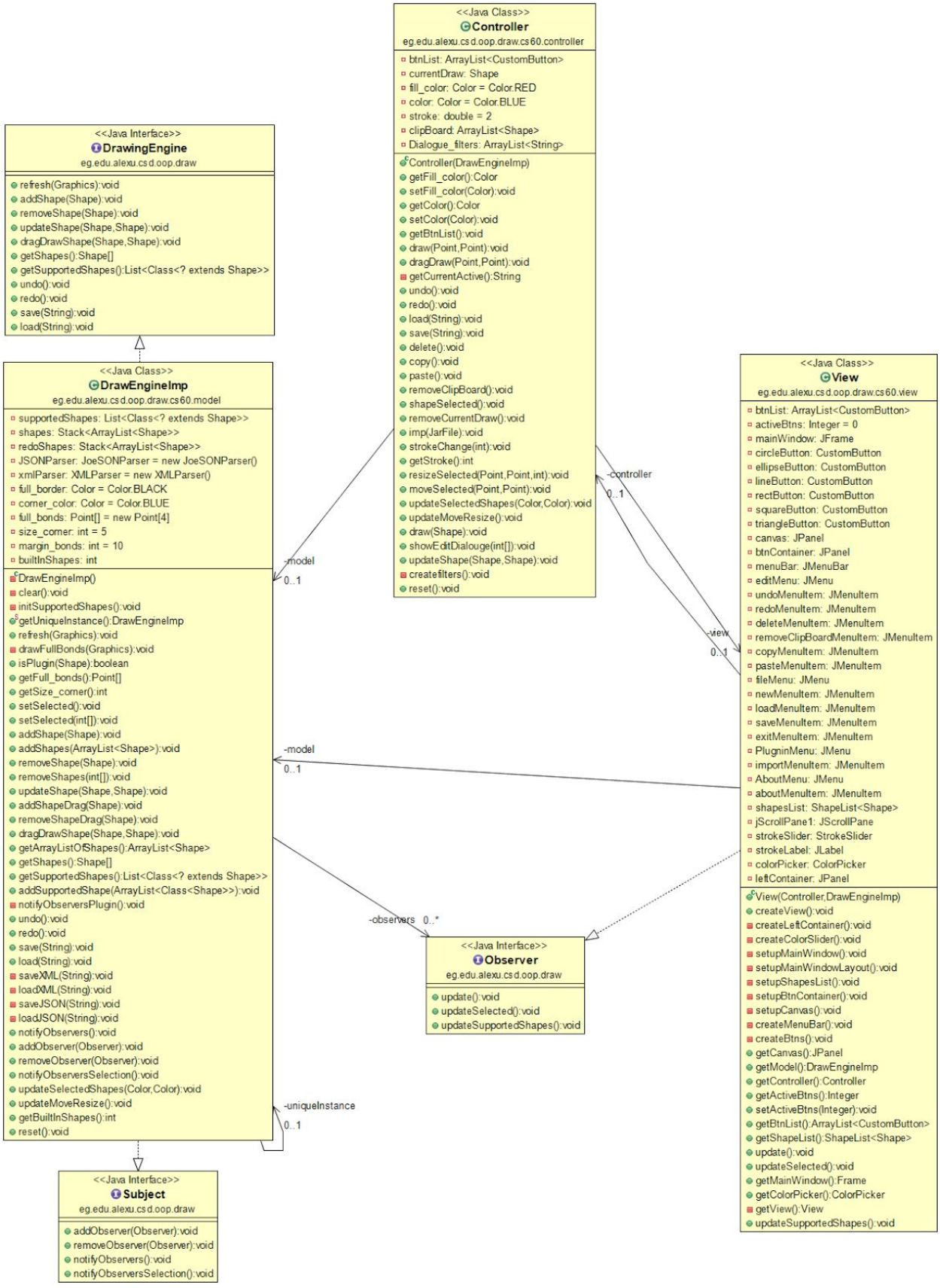
Youssef Ali (73)

INTRODUCTION

A desktop application for drawing geometric shapes with many features such as coloring, resizing, moving, copying and pasting, saving and loading and ability to add extensions at runtime. It was implemented in Java using Swing framework for the GUI.

UML DIAGRAMS




<<Java Class>>
Controller
eg.edu.alexu.csd.oop.draw.cs60.controller
btnList: ArrayList<CustomButton>
currentDraw: Shape
fill_color: Color = Color.RED
color: Color = Color.BLUE
stroke: double = 2
clipBoard: ArrayList<Shape>
Dialogue_filters: ArrayList<String>
Controller(DrawEngineImp)
getFill_color():Color
setFill_color(Color):void
getColor():Color
setColor(Color):void
getBtnList():void
draw(Point,Point):void
dragDraw(Point,Point):void
getCurrentActive():String
undo():void
redo():void
load(String):void
save(String):void
delete():void
copy():void
paste():void
removeClipBoard():void
shapeSelected():void
removeCurrentDraw():void
imp(JarFile):void
strokeChange(int):void
getStroke():int
resizeSelected(Point,Point,int):void
moveSelected(Point,Point):void
updateSelectedShapes(Color,Color):void
updateMoveResize():void
draw(Shape):void
showEditDialouge(int[]):void
updateShape(Shape,Shape):void
createfilters():void
reset():void
<<Java Interface>>
DrawingEngine
eg.edu.alexu.csd.oop.draw
refresh(Graphics):void
addShape(Shape):void
removeShape(Shape):void
updateShape(Shape,Shape):void
dragDrawShape(Shape,Shape):void
getShapes():Shape[]
getSupportedShapes():List<Class<? extends Shape>>
undo():void
redo():void
save(String):void
load(String):void
<<Java Class>>
DrawEngineImp
eg.edu.alexu.csd.oop.draw.cs60.model
supportedShapes: List<Class<? extends Shape>>
shapes: Stack<ArrayList<Shape>>
redoShapes: Stack<ArrayList<Shape>>
JSONParser: JoeSONParser = new JoeSONParser()
xmlParser: XMLParser = new XMLParser()
full_border: Color = Color.BLACK
corner_color: Color = Color.BLUE
full_bonds: Point[] = new Point[4]
size_corner: int = 5
margin_bonds: int = 10
builtInShapes: int
DrawEngineImp()
clear():void
initSupportedShapes():void
getUniqueInstance():DrawEngineImp
refresh(Graphics):void
drawFullBonds(Graphics):void
isPlugin(Shape):boolean
getFull_bonds():Point[]
getSize_corner():int
setSelected():void
setSelected(int[]):void
addShape(Shape):void
addShapes(ArrayList<Shape>):void
removeShape(Shape):void
removeShapes(int[]):void
updateShape(Shape,Shape):void
addShapeDrag(Shape):void
removeShapeDrag(Shape):void
dragDrawShape(Shape,Shape):void
getArrayListOfShapes():ArrayList<Shape>
getShapes():Shape[]
getSupportedShapes():List<Class<? extends Shape>>
addSupportedShape(ArrayList<Class<Shape>>):void
notifyObserversPlugin():void
undo():void
redo():void
save(String):void
load(String):void
saveXML(String):void
loadXML(String):void
saveJSON(String):void
loadJSON(String):void
notifyObservers():void
addObserver(Observer):void
removeObserver(Observer):void
notifyObserversSelection():void
updateSelectedShapes(Color,Color):void
updateMoveResize():void
getBuiltInShapes():int
reset():void
<<Java Class>>
View
eg.edu.alexu.csd.oop.draw.cs60.view
btnList: ArrayList<CustomButton>
activeBtns: Integer = 0
mainWindow: JFrame
circleButton: CustomButton
ellipseButton: CustomButton
lineButton: CustomButton
rectButton: CustomButton
squareButton: CustomButton
triangleButton: CustomButton
canvas: JPanel
btnContainer: JPanel
menuBar: JMenuBar
editMenu: JMenu
undoMenuItem: JMenuItem
redoMenuItem: JMenuItem
deleteMenuItem: JMenuItem
removeClipBoardMenuItem: JMenuItem
copyMenuItem: JMenuItem
pasteMenuItem: JMenuItem
fileMenu: JMenu
newMenuItem: JMenuItem
loadMenuItem: JMenuItem
saveMenuItem: JMenuItem
exitMenuItem: JMenuItem
PluginMenu: JMenu
importMenuItem: JMenuItem
AboutMenu: JMenu
aboutMenuItem: JMenuItem
shapesList: ShapeList<Shape>
jScrollPane1: JScrollPane
strokeSlider: StrokeSlider
strokeLabel: JLabel
colorPicker: ColorPicker
leftContainer: JPanel
View(Controller,DrawEngineImp)
createView():void
createLeftContainer():void
createColorSlider():void
setupMainWindow():void
setupMainWindowLayout():void
setupShapesList():void
setupBtnContainer():void
setupCanvas():void
createMenuBar():void
createBtns():void
getCanvas():JPanel
getModel():DrawEngineImp
getController():Controller
getActiveBtns():Integer
setActiveBtns(Integer):void
getBtnList():ArrayList<CustomButton>
getShapeList():ShapeList<Shape>
update():void
updateSelected():void
getMainWindow():Frame
getColorPicker():ColorPicker
getView():View
updateSupportedShapes():void
<<Java Interface>>
Observer
eg.edu.alexu.csd.oop.draw
update():void
updateSelected():void
updateSupportedShapes():void
<<Java Interface>>
Subject
eg.edu.alexu.csd.oop.draw
addObserver(Observer):void
removeObserver(Observer):void
notifyObservers():void
notifyObserversSelection():void
-controller
0..1
-model
0..1
-view
0..1
-model
0..1
-observers
0..*
-uniqueInstance
0..1

OUR DESIGN

The Program Consists Primarily of 3 Parts, Model, View and Controller, According to MVC architectural pattern.
Different Design Patterns such as Factory, Singleton and Observer were used.

- **View**

View was designed almost as separate modules such as

- ***CustomButton*** : an Extension to the *JButton* that has two 2 States, active and non_active, for allowing user to draw different shapes.
- ***ShapeList*** : an Extension to the *JList* that allowed multiple selection , dynamic changing elements according to the model and interacting with different shapes drawn.
- ***ColorPicker*** : an Extension to the *JButton* that works as a preview for the current used Colors and allows for changing color through a colorchooser.
- ***CreateDialogue*** : an Extension to the *JDialogue* that get constructed dynamically to allow dynamic drawing for different shapes according to their properties.
- ***Canvas*** : an Extension to the *JPanel* that is used as a drawing area for shapes.
- ***Save, Load and Plugin Choosers*** : an Extension to the *JFileChooser* that allowed file fetching in the importing plugin or loading and path fetching in case of saving.
- ***StrokeSlider*** : an Extension to the *JSlider* that allows the user to change stroke level.

Different view components are dynamically changing based on the model Thanks to Observer Design pattern and MVC.

- **Controller**

Controller is the MiddleMan between the View and Model in our case the *DrawingEngineImp* as updating the Model whenever the view is affecting as well as changing the view in several situations.

- **Model**

Model in our case is Representing by the *DrawingEngine* Implementation, it is responsible about dealing with the different Shapes updating, removing and adding different shapes.

SNAPSHOTS

Fig.1 : Main Window

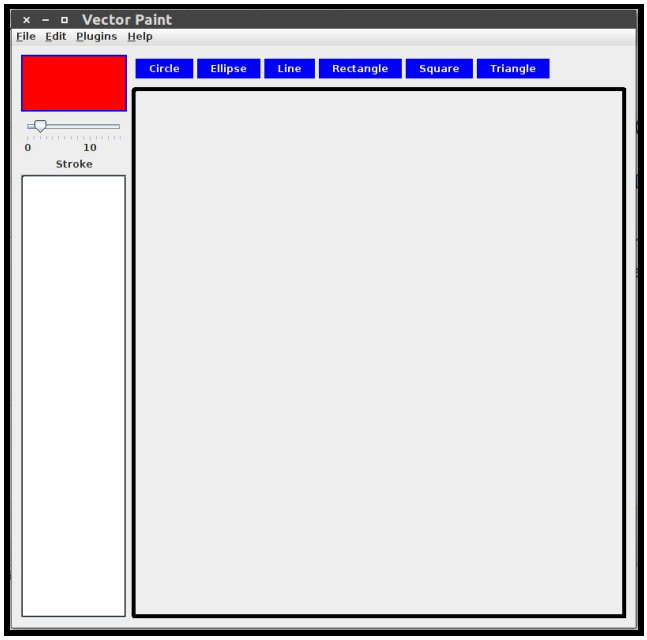


Fig.2 : Drawing Example

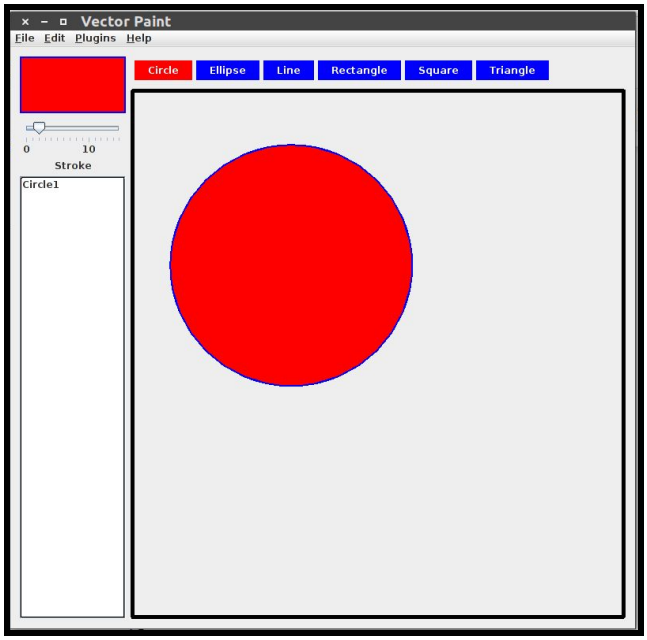


Fig.3 : Selecting Feature

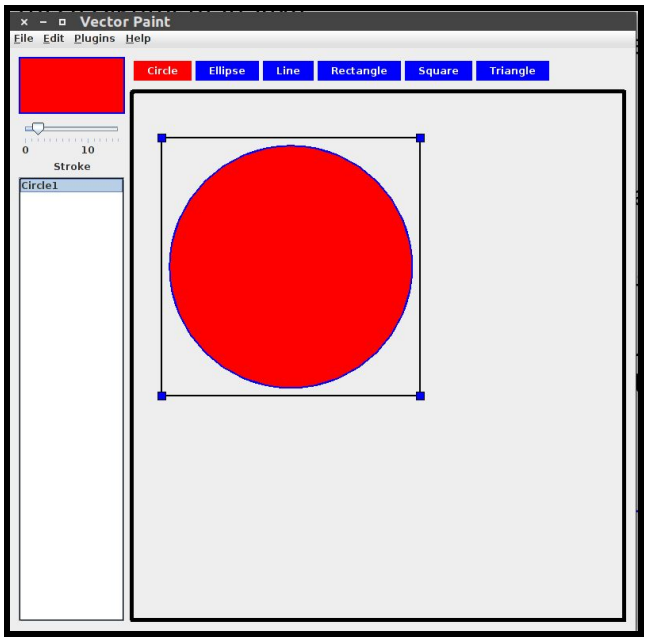


Fig.4 : Moving Feature

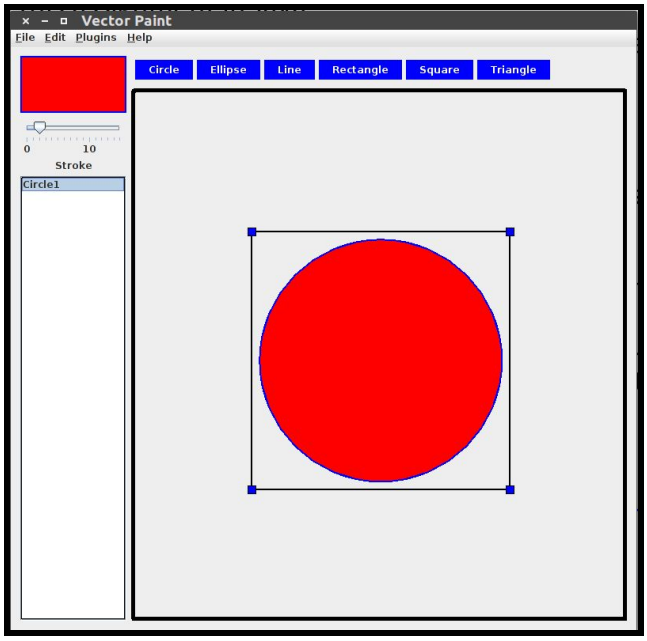


Fig.5 : Resizing Feature

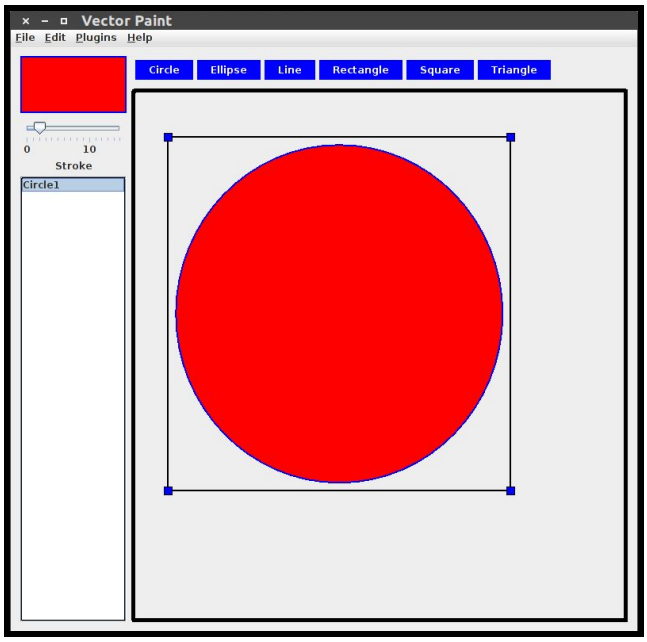


Fig.6 : Copy & Paste Feature

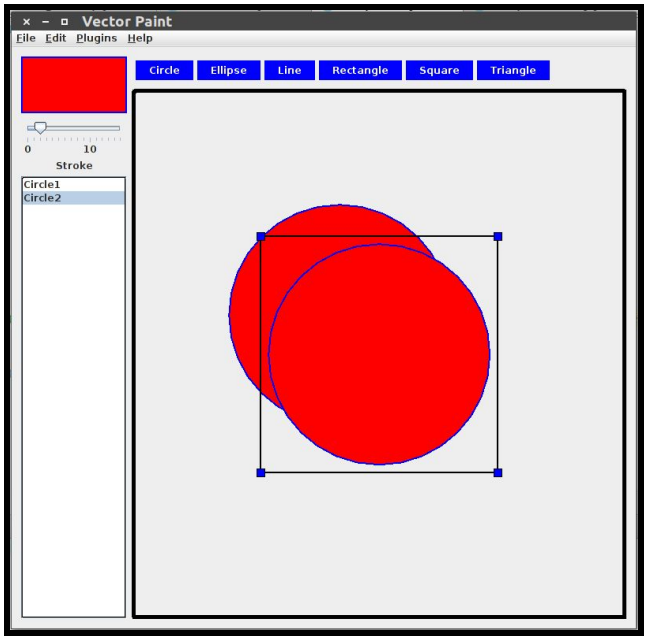


Fig.7 : Save File Chooser

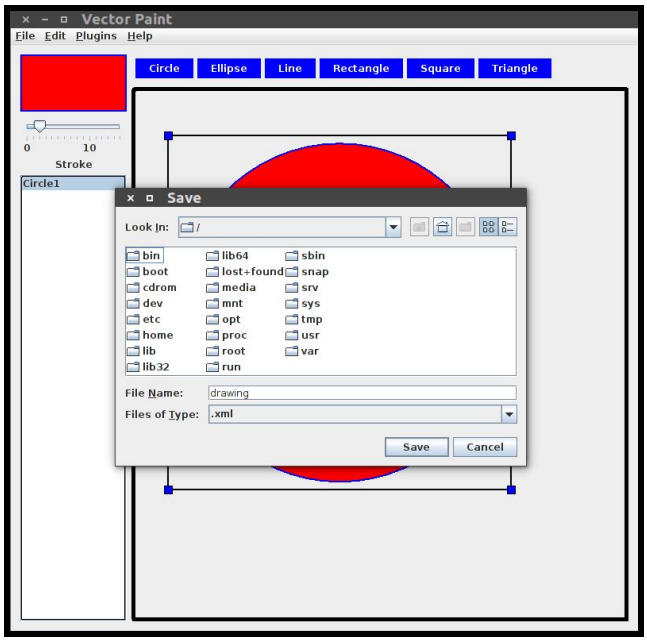


Fig.8 : Load File Chooser

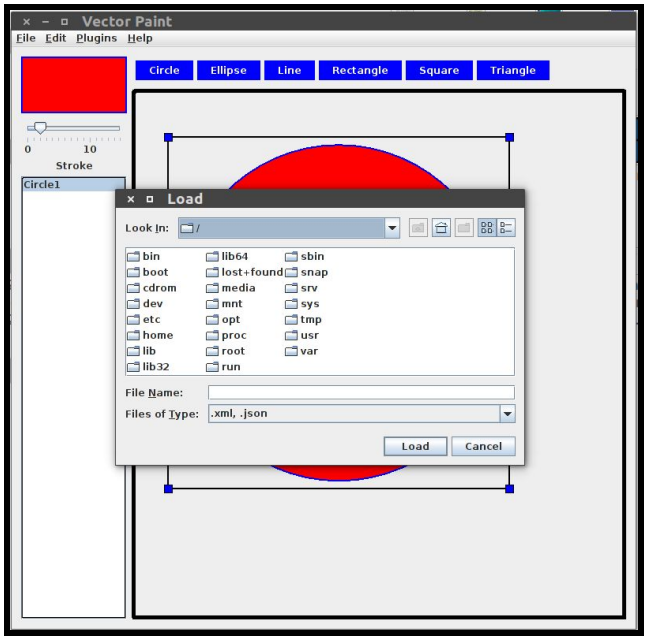


Fig.9 : Plugin File Chooser

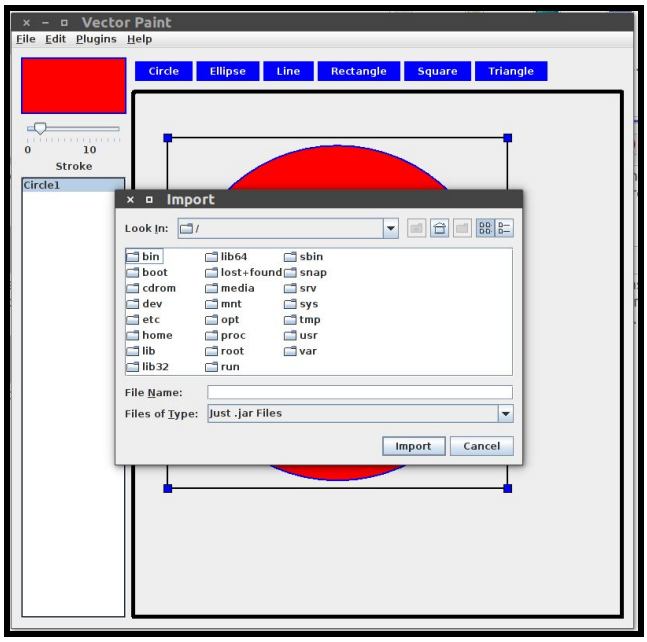


Fig.10 : Plugin Button Added

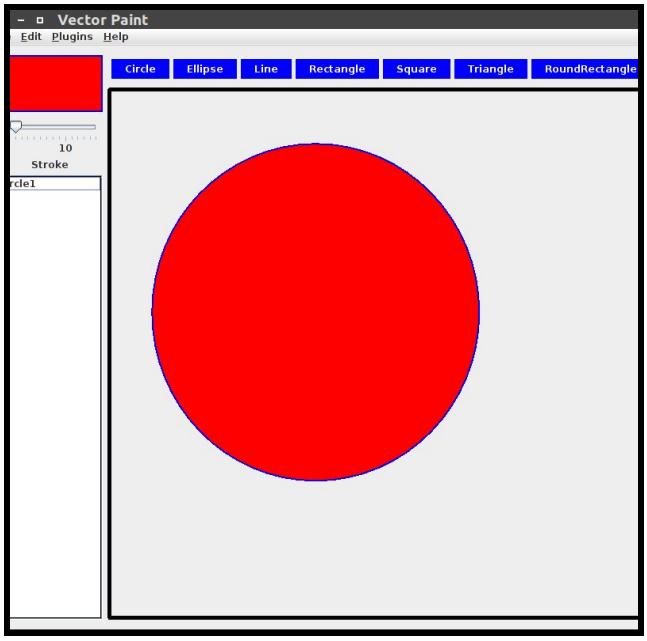
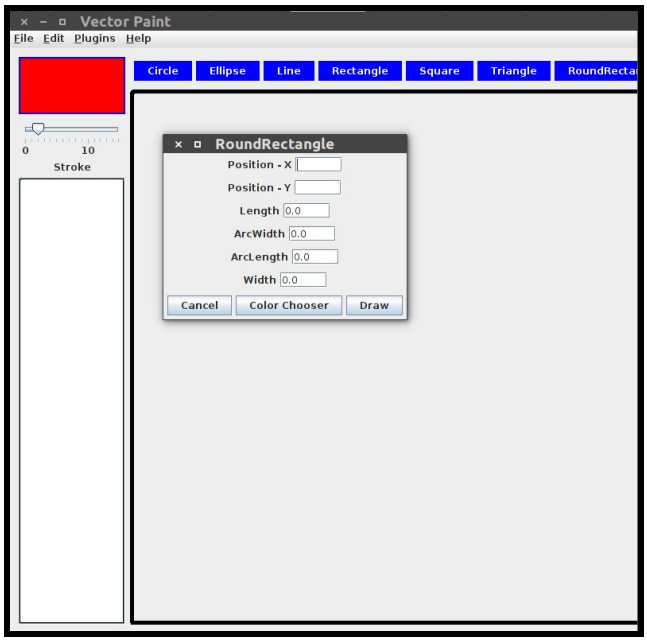


Fig.11 : Drawing An Imported Shape Using a Dialog Box



USER GUIDE

- **How to draw:**
  1. Select a shape type from the shape types list.
  2. Press and drag your mouse inside the drawing area.
  3. Release your mouse and the shape is drawn as in Fig. 2.
- **How to move:**
  1. Deselect the selected shape type if any.
  2. Select any shape(s) from the shape list in the right as in Fig. 3.
  3. Move by dragging the mouse as in Fig. 4.
- **How to resize:**
  1. Deselect the selected shape type if any.
  2. Select any shape(s) from the shape list in the right as in Fig. 3.
  3. Resize using the four corners that appears as in Fig. 5.
- **How to undo:**
  1. From the menu bar open Edit menu and select Undo or just click CTRL+Z.
- **How to redo:**
  1. From the menu bar open Edit menu and select Redo or just click CTRL+Y.

- How to Copy:
    1. Select any shape(s) from the shape list in the right as in Fig. 3.
    2. From the menu bar open File menu and select Copy Shape(s) or just click SHIFT+C.
- How to Paste:
    1. From the menu bar open File menu and select Paste Shape(s) or just click SHIFT+V.
    2. The pasted shape(s) is/are added to the shape list as in Fig. 6.
- How to Save:
    1. From the menu bar open File menu and select Save or just click CTRL+S.
    2. From the window that appears select path of the file to be saved as in Fig. 7.
    3. Select the file type. You can save it in XML or JSON format.
    4. Hit the save button.
- How to Load:
    1. From the menu bar open File menu and select Load.
    2. From the window that appears select the file to be loaded as in Fig. 8.
    3. You can load just XML or JSON files that was have been already saved by the application.
    4. Hit the load button.
- How to import a plugin shape:
    1. From the menu bar open Plugins menu and select Import or just click CTRL+I.
    2. From the window that appears select the JAR file to be loaded as in Fig. 9.
    3. You can import just JAR files that contain a shape class(es).
    4. Hit the import button.
- How to draw a loaded plugin shape:
    1. After importing a plugin shape, it is added to the shape types list as in Fig. 10.
    2. Select it from shape types list.
    3. Fill in the dialog box that appears as in Fig. 11.
    4. Set color by hitting the color chooser button.
    5. Hit the draw button.
- How to change the drawing color:
    1. Click on the color chooser button on the upper left corner.
    2. To Set a filling color, choose any color and hit select color button.
    3. To Set a border color, choose any color and hit select border button.
- How to change the drawing stroke thickness:
    1. Choose the desired stroke by controlling the slider in the up left.
    2. You can see the current chosen stroke applied on the color chooser button above of the slider.
- How to change the color of a drawn shape:
    1. Select any shape(s) from the shape list in the right as in Fig. 3.
    2. Right Click on them or click F from the keyboard.

3. A color picker window appears.
4. To Set a filling color, choose any color and hit select color button.
5. To Set a border color, choose any color and hit select border button.

- ### How to change the properties of a drawn shape:
    1. Select any shape(s) from the shape list in the right as in Fig. 3.
    2. Click D from the keyboard.
    3. A dialog box appears to set the properties of every selected shape.