VECTOR DESIGN TOOL

Jessica Williams	n10194363
Jacob Kraut	n10282653
William Daley	n9960961

Statement of Completeness

Basic Functionality

Functionality	Implemented
Saving VEC Images	✓
Loading VEC Images	✓
Plot	✓
Line	✓
Rectangle	✓
Ellipse	✓
Polygon	✓
Undo	✓

Advanced Functionality

Functionality	Implemented
Grid	✓
Rainbow Button	✓

Statement of Contribution

Name	**Contribution**
Jessica Williams	Report, GUI, Junit Tests, JavaDoc
Jacob Kraut	Contributed to all classes, Junit Tests
William Daley	I/O Class, Contributed to all classes besides GUI, JavaDoc, advanced functionality

Agile Software Development Process

Our team did not make use of test-driven development but we engaged in pair programming, sprint planning, iteration and weekly stand-ups.

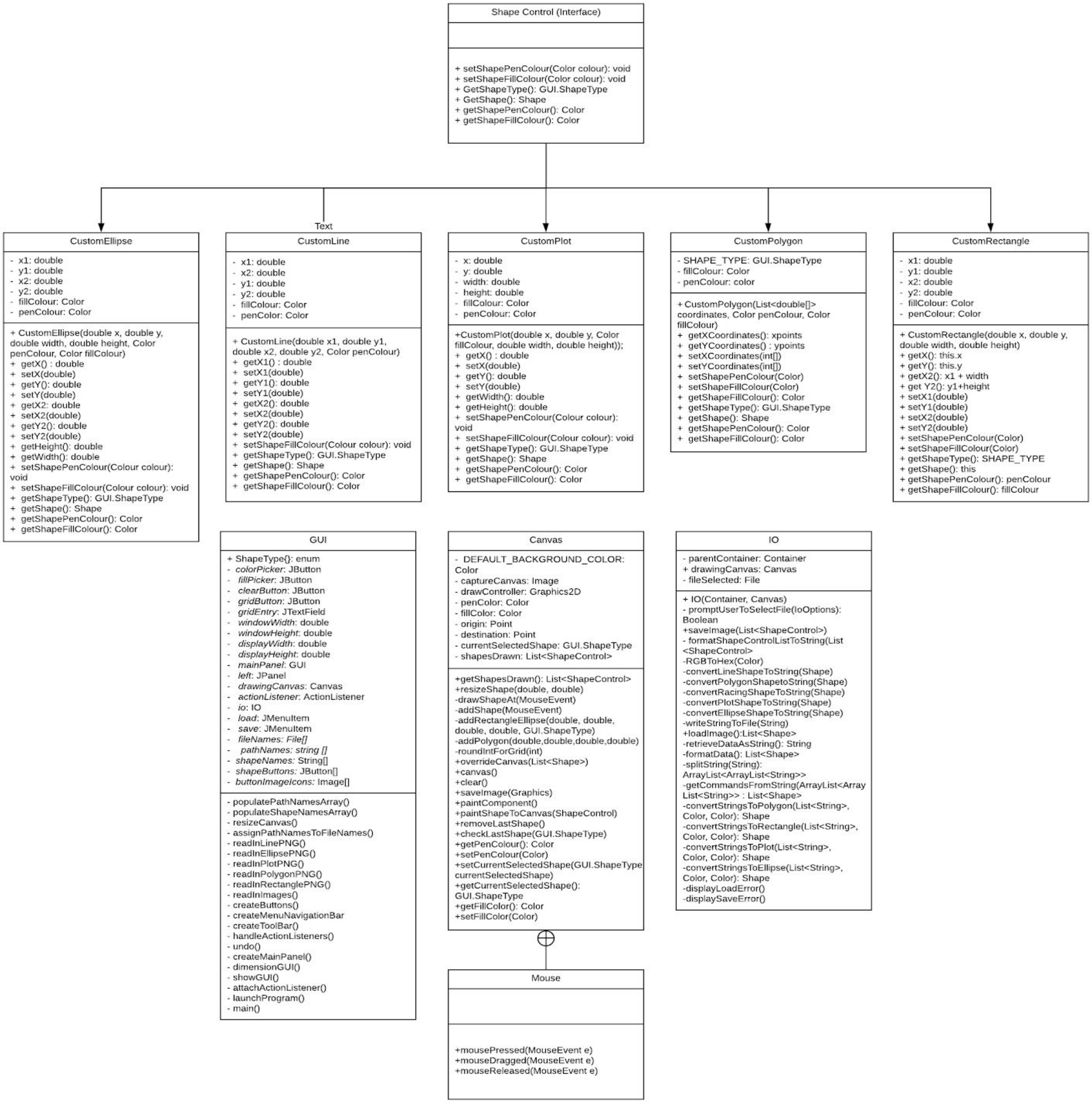
Each sprint was a week long and involved a short planning phase for what we wanted to get done and a reflection on what we had gotten done. We also agreed to a verbal ‘social contract’ and agreed that we would start this assignment ASAP and meet once a week until the assignment is done.

Agile methodology recommends daily stand-ups but we found it wasn’t feasible for us as we have other university and work commitments. Instead we organised a weekly standup on Saturday or Sunday mornings. We would discuss what we said we’d do, what we did and what we are going to do in our next sprint.

Pair programming was used heavily during our weekend work, which made up the majority of our work. As you can see in our commit history, a lot of commits come from one person for a length of time and then switches to another. This is us switching the ‘driver’ and ‘navigator’ role.

Software Architecture

Our program allows for extensibility very easily. If a new shape button was needed, a new class could be added and an extra button in the GUI. There would be no redesigning of the architecture.


Shape Control (Interface)
+ setShapePenColour(Color colour): void
+ setShapeFillColour(Color colour): void
+ GetShapeType(): GUI.ShapeType
+ GetShape(): Shape
+ getShapePenColour(): Color
+ getShapeFillColour(): Color
Text
CustomEllipse
- x1: double
- y1: double
- x2: double
- y2: double
- fillColour: Color
- penColour: Color
+ CustomEllipse(double x, double y, double width, double height, Color penColour, Color fillColour)
+ getX() : double
+ setX(double)
+ getY(): double
+ setY(double)
+ getX2: double
+ setX2(double)
+ getY2(): double
+ setY2(double)
+ getHeight(): double
+ getWidth(): double
+ setShapePenColour(Colour colour): void
+ setShapeFillColour(Colour colour): void
+ getShapeType(): GUI.ShapeType
+ getShape(): Shape
+ getShapePenColour(): Color
+ getShapeFillColour(): Color
CustomLine
- x1: double
- x2: double
- y1: double
- y2: double
- fillColour: Color
- penColor: Color
+ CustomLine(double x1, double y1, double x2, double y2, Color penColour)
+ getX1() : double
+ setX1(double)
+ getY1(): double
+ setY1(double)
+ getX2(): double
+ setX2(double)
+ getY2(): double
+ setY2(double)
+ setShapeFillColour(Colour colour): void
+ getShapeType(): GUI.ShapeType
+ getShape(): Shape
+ getShapePenColour(): Color
+ getShapeFillColour(): Color
CustomPlot
- x: double
- y: double
- width: double
- height: double
- fillColour: Color
- penColour: Color
+CustomPlot(double x, double y, Color fillColour, double width, double height));
+ getX() : double
+ setX(double)
+ getY(): double
+ setY(double)
+ getWidth(): double
+ getHeight(): double
+ setShapePenColour(Colour colour): void
+ setShapeFillColour(Colour colour): void
+ getShapeType(): GUI.ShapeType
+ getShape(): Shape
+ getShapePenColour(): Color
+ getShapeFillColour(): Color
CustomPolygon
- SHAPE_TYPE: GUI.ShapeType
- fillColour: Color
- penColour: color
+ CustomPolygon(List<double[]> coordinates, Color penColour, Color fillColour)
+ getXCoordinates(): xpoints
+ getYCoordinates() : ypoints
+ setXCoordinates(int[])
+ setYCoordinates(int[])
+ setShapePenColour(Color)
+ setShapeFillColour(Color)
+ getShapeFillColour(): Color
+ getShapeType(): GUI.ShapeType
+ getShape(): Shape
+ getShapePenColour(): Color
+ getShapeFillColour(): Color
CustomRectangle
- x1: double
- y1: double
- x2: double
- y2: double
- fillColour: Color
- penColour: Color
+ CustomRectangle(double x, double y, double width, double height)
+ getX(): this.x
+ getY(): this.y
+ getX2(): x1 + width
+ get Y2(): y1+height
+ setX1(double)
+ setY1(double)
+ setX2(double)
+ setY2(double)
+ setShapePenColour(Color)
+ setShapeFillColour(Color)
+ getShapeType(): SHAPE_TYPE
+ getShape(): this
+ getShapePenColour(): penColour
+ getShapeFillColour(): fillColour
GUI
+ ShapeType{}: enum
- colorPicker: JButton
- fillPicker: JButton
- clearButton: JButton
- gridButton: JButton
- gridEntry: JTextField
- windowWidth: double
- windowHeight: double
- displayWidth: double
- displayHeight: double
- mainPanel: GUI
- left: JPanel
- drawingCanvas: Canvas
- actionListener: ActionListener
- io: IO
- load: JMenuItem
- save: JMenuItem
- fileNames: File[]
- pathNames: string []
- shapeNames: String[]
- shapeButtons: JButton[]
- buttonImageIcons: Image[]
- populatePathNamesArray()
- populateShapeNamesArray()
- resizeCanvas()
- assignPathNamesToFileNames()
- readInLinePNG()
- readInEllipsePNG()
- readInPlotPNG()
- readInPolygonPNG()
- readInRectanglePNG()
- readInImages()
- createButtons()
- createMenuNavigationBar
- createToolBar()
- handleActionListeners()
- undo()
- createMainPanel()
- dimensionGUI()
- showGUI()
- attachActionListener()
- launchProgram()
- main()
Canvas
- DEFAULT_BACKGROUND_COLOR: Color
- captureCanvas: Image
- drawController: Graphics2D
- penColor: Color
- fillColor: Color
- origin: Point
- destination: Point
- currentSelectedShape: GUI.ShapeType
- shapesDrawn: List<ShapeControl>
+getShapesDrawn(): List<ShapeControl>
+resizeShape(double, double)
-drawShapeAt(MouseEvent)
-addShape(MouseEvent)
-addRectangleEllipse(double, double, double, double, GUI.ShapeType)
-addPolygon(double,double,double,double)
-roundIntForGrid(int)
+overrideCanvas(List<Shape>)
+canvas()
+clear()
+saveImage(Graphics)
+paintComponent()
+paintShapeToCanvas(ShapeControl)
+removeLastShape()
+checkLastShape(GUI.ShapeType)
+getPenColour(): Color
+setPenColour(Color)
+setCurrentSelectedShape(GUI.ShapeType currentSelectedShape)
+getCurrentSelectedShape(): GUI.ShapeType
+getFillColor(): Color
+setFillColor(Color)
Mouse
+mousePressed(MouseEvent e)
+mouseDragged(MouseEvent e)
+mouseReleased(MouseEvent e)
IO
- parentContainer: Container
+ drawingCanvas: Canvas
- fileSelected: File
+ IO(Container, Canvas)
- promptUserToSelectFile(IoOptions): Boolean
+saveImage(List<ShapeControl>)
- formatShapeControlListToString(List <ShapeControl>
-RGBToHex(Color)
-convertLineShapeToString(Shape)
-convertPolygonShapetoString(Shape)
-convertRacingShapeToString(Shape)
-convertPlotShapeToString(Shape)
-convertEllipseShapeToString(Shape)
-writeStringToFile(String)
+loadImage():List<Shape>
-retrieveDataAsString(): String
-formatData(): List<Shape>
-splitString(String): ArrayList<ArrayList<String>>
-getCommandsFromString(ArrayList<ArrayList<String>> : List<Shape>
-convertStringsToPolygon(List<String>, Color, Color): Shape
-convertStringsToRectangle(List<String>, Color, Color): Shape
-convertStringsToPlot(List<String>, Color, Color): Shape
-convertStringsToEllipse(List<String>, Color, Color): Shape
-displayLoadError()
-displaySaveError()

Object Oriented Programming Principles

Abstraction

Demonstrated by the use of the ‘Shape Control’ class. Shape Control is an interface that contains all the common methods between the shapes. Abstraction is demonstrated by this interface because it’s displaying the essential methods and properties of each shape object.

Encapsulation

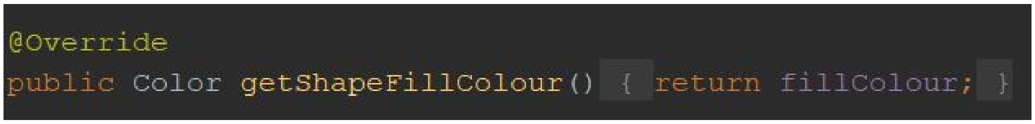
By using classes with private variables, our program is demonstrating encapsulation. Our classes are enclosing all of the information about each object.

Inheritance

Our program demonstrates inheritance by extending from java libraries (Example: ShapeControl extends Shape) and implementing from Shape (Example: CustomPolygon extends polygon implements ShapeControl).

Polymorphism

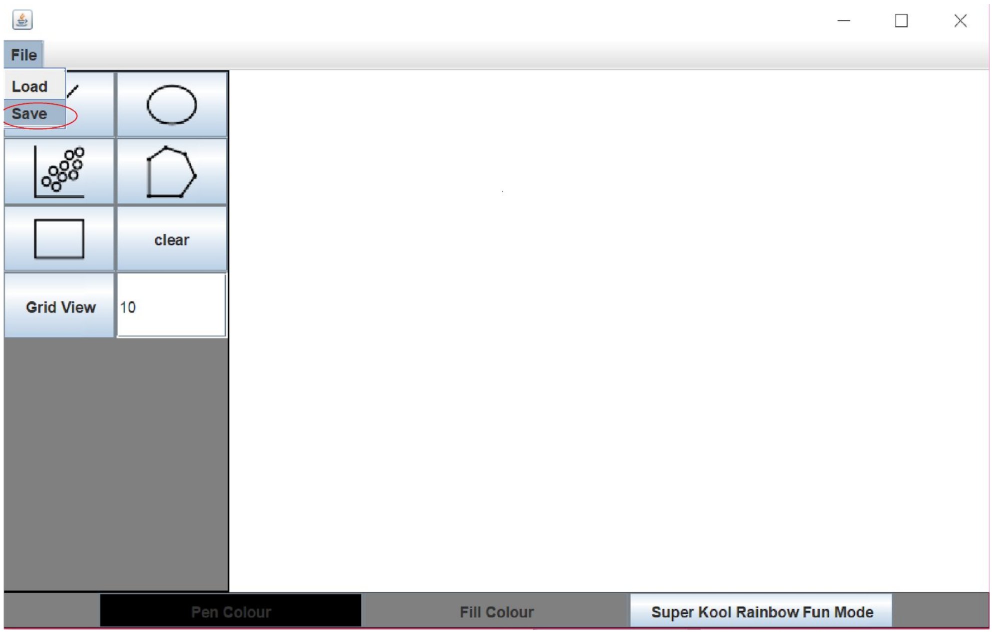
This is demonstrated by overriding methods and variables from the ShapeControl interface in each custom shape class we have. For example:



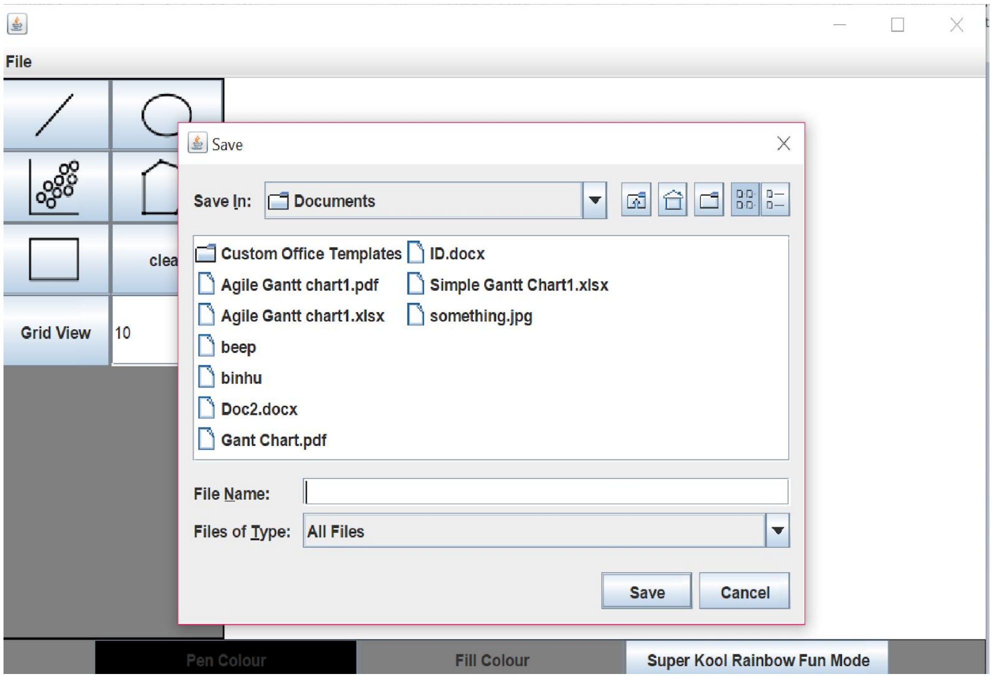
```
@Override
public Color getShapeFillColour() { return fillColour; }
```

How to Use Our Software

Saving VEC Images

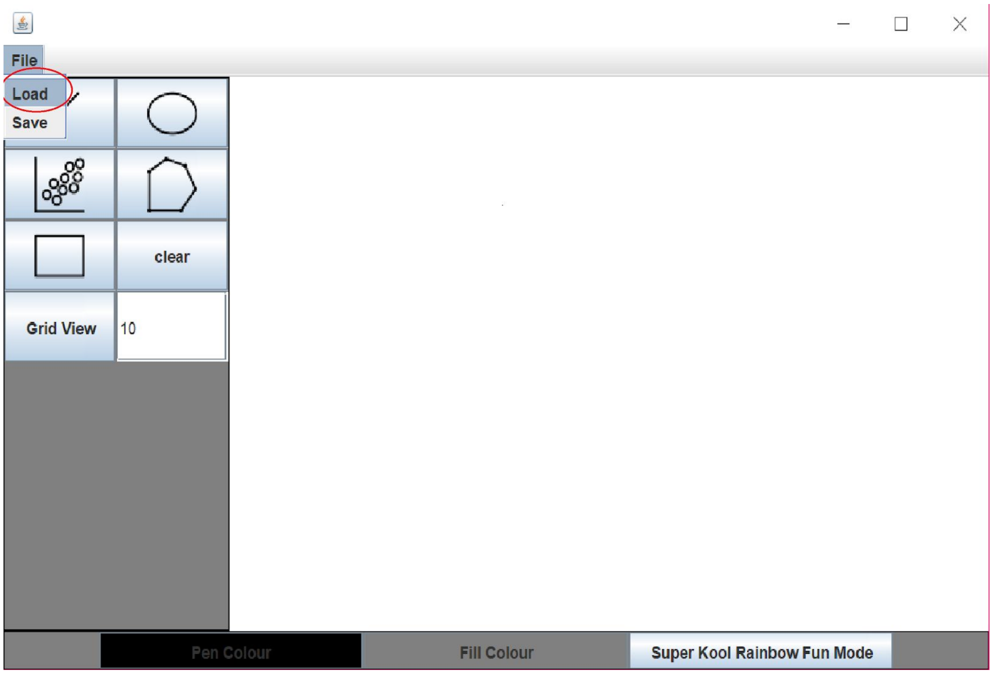


To Save a VEC image, select “File” and then “Save”

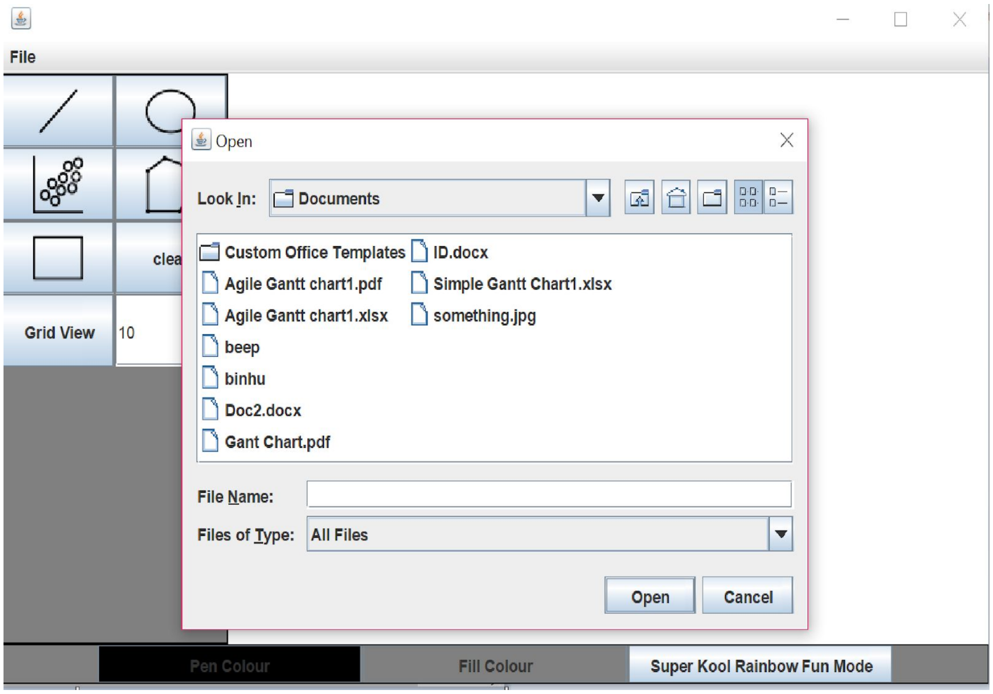


Write your file name, and select “Save” - you now have saved your VEC image.

Loading VEC Images



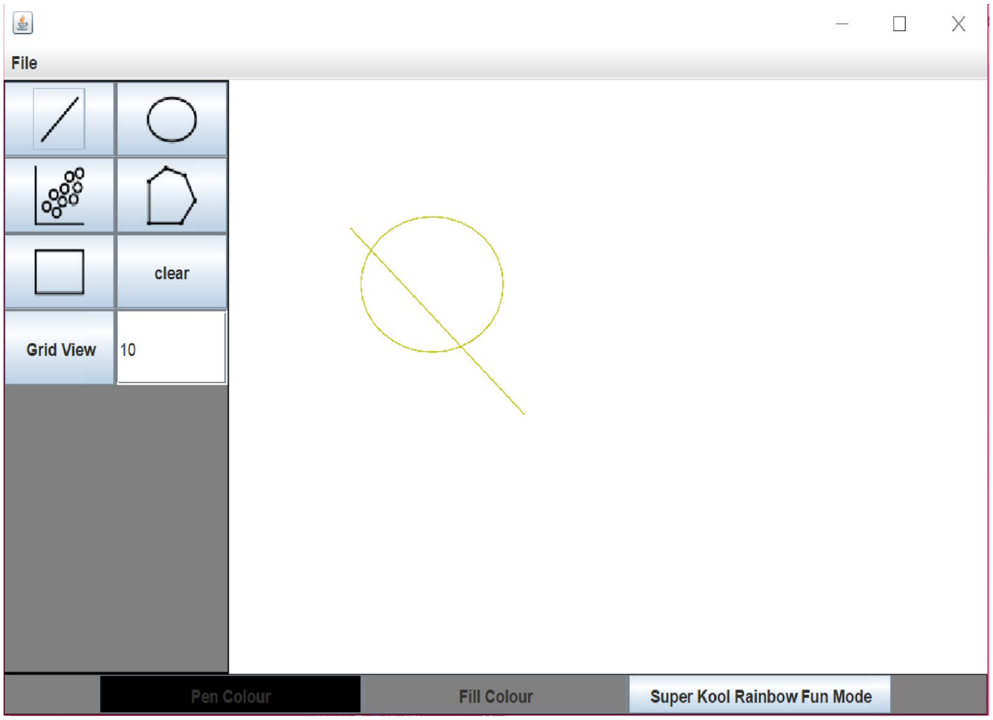
To Load a VEC Image into the program, go to “File” and select “Load”



Look In: Documents

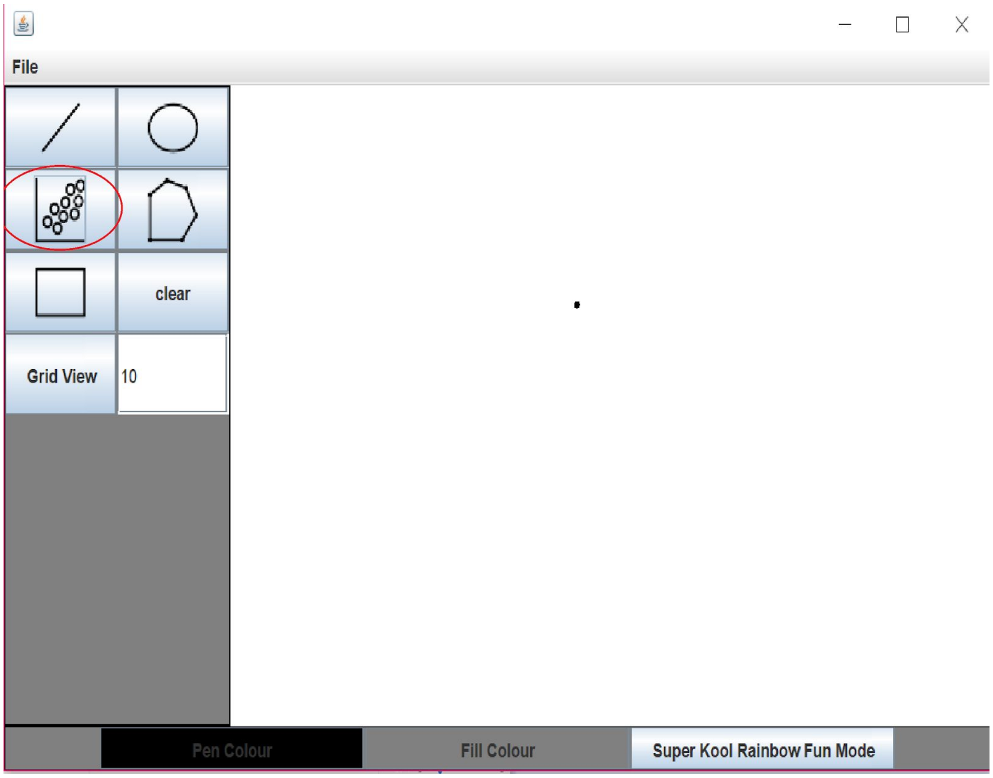
Files of Type: All Files

This file explorer will open up, select your VEC file and then double click or press “Open”.



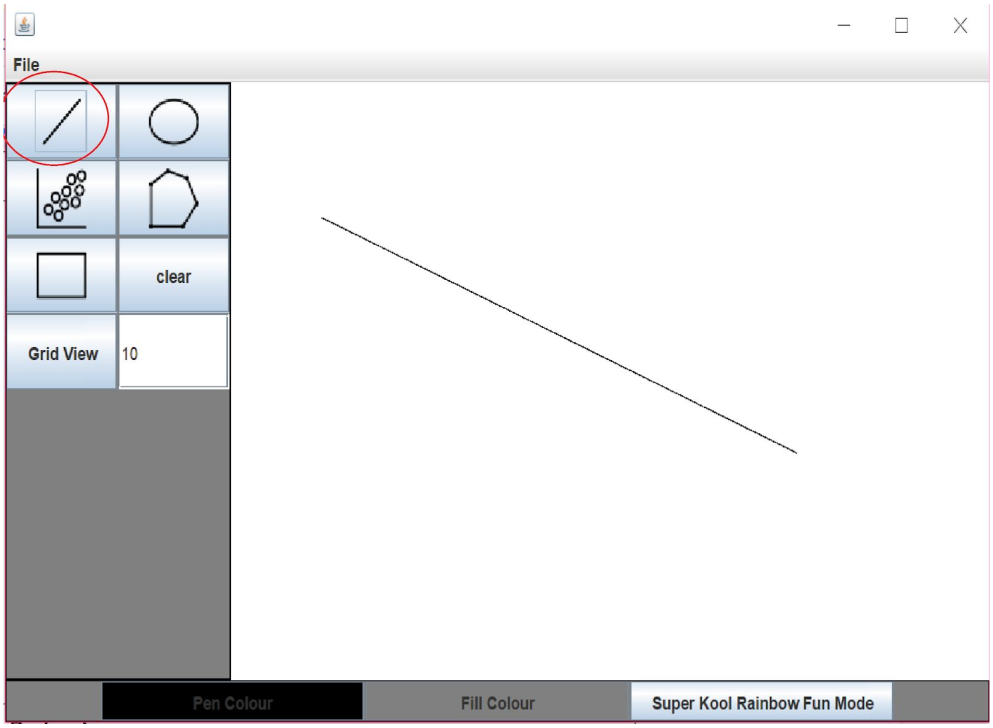
You can now see your VEC image loaded into the canvas.

Plot



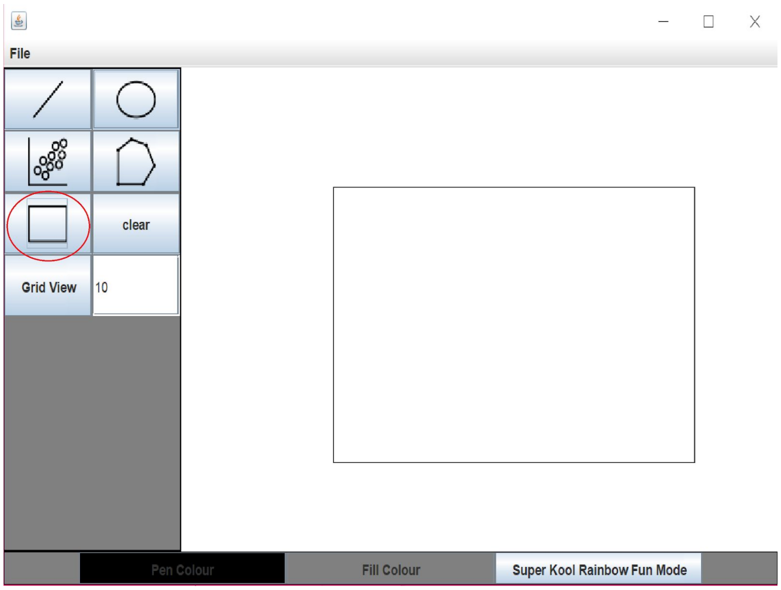
To place a plot on the canvas, select the button circled in red and click and drag on the canvas.

Line



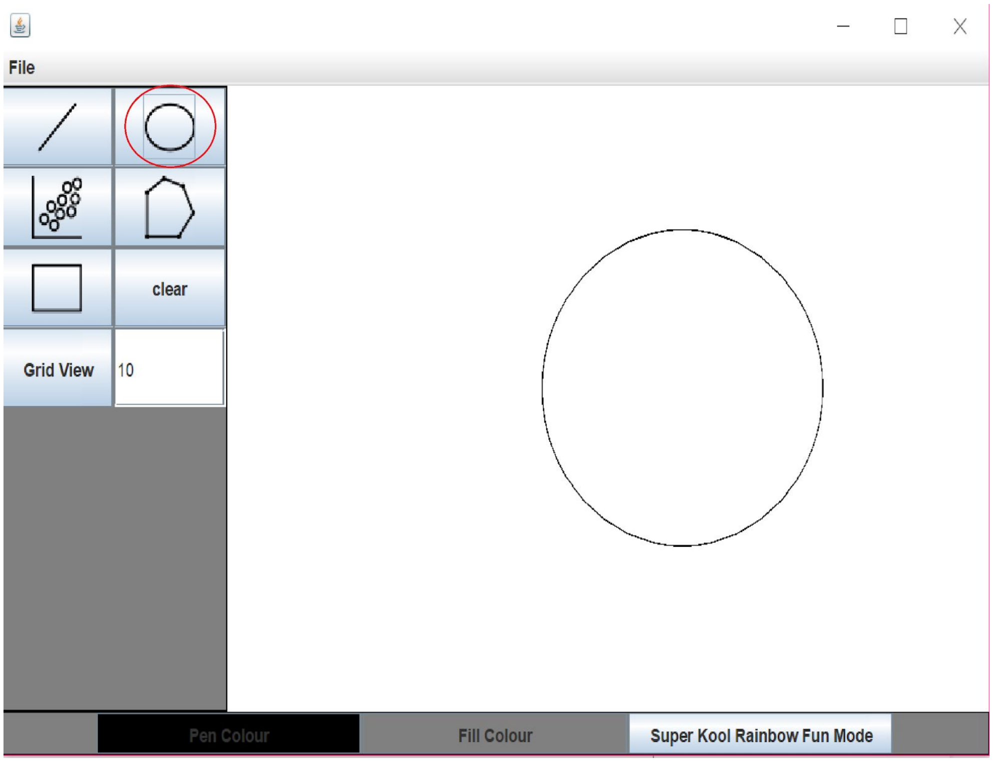
To place a Line on the canvas, select the button circled in red and click and drag on the canvas..

Rectangle



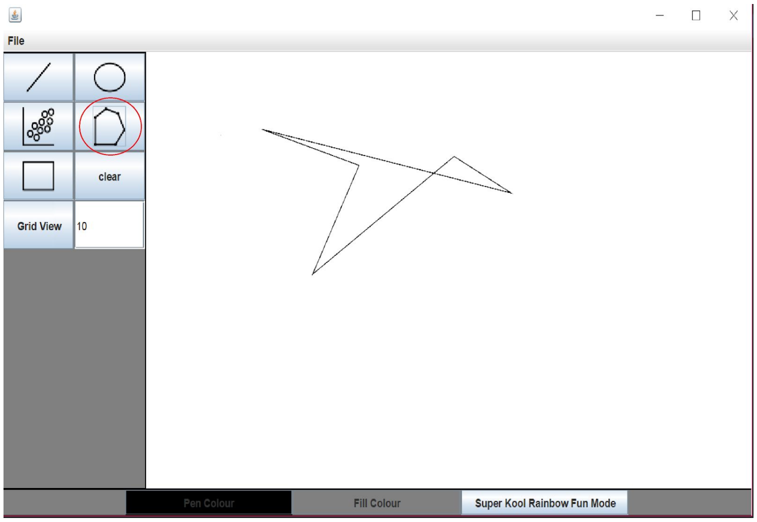
To place a rectangle on the canvas, select the button circled in red and click and drag on the canvas.

Ellipse



To place an ellipse on the canvas, select the button circled in red and click and drag on the canvas.

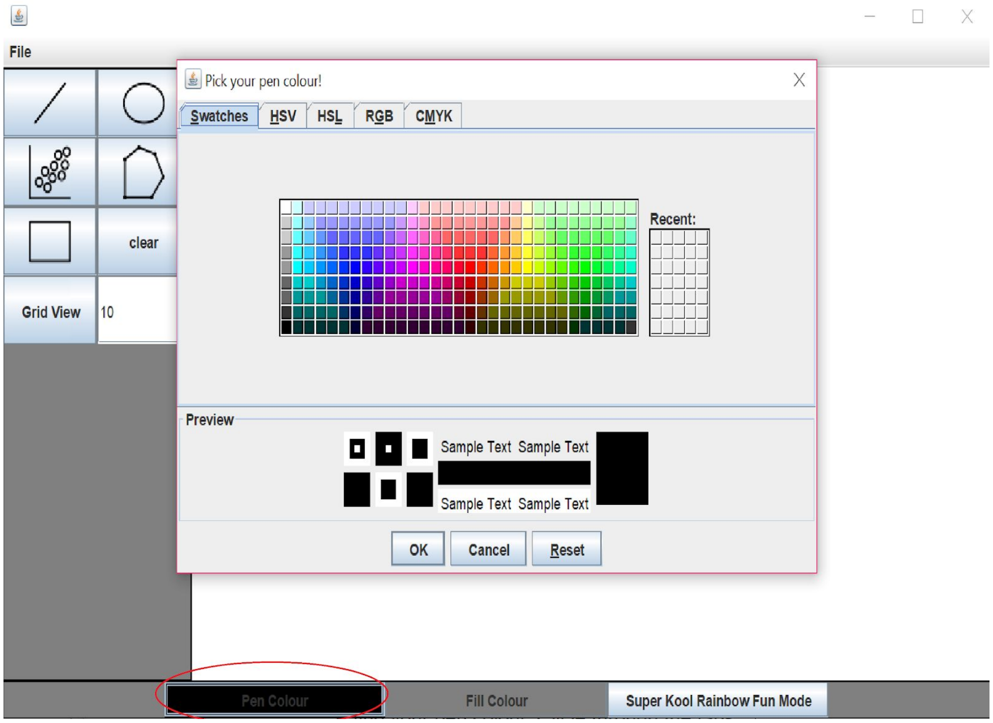
Polygon



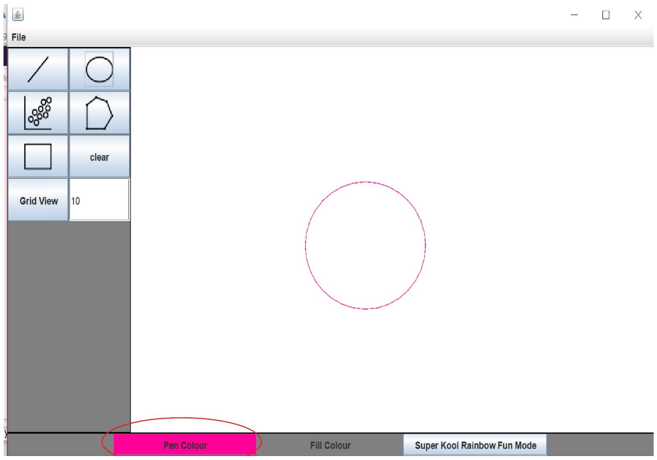
To place a polygon on the canvas, select the button circled in red and click and drag on the canvas.

Undo

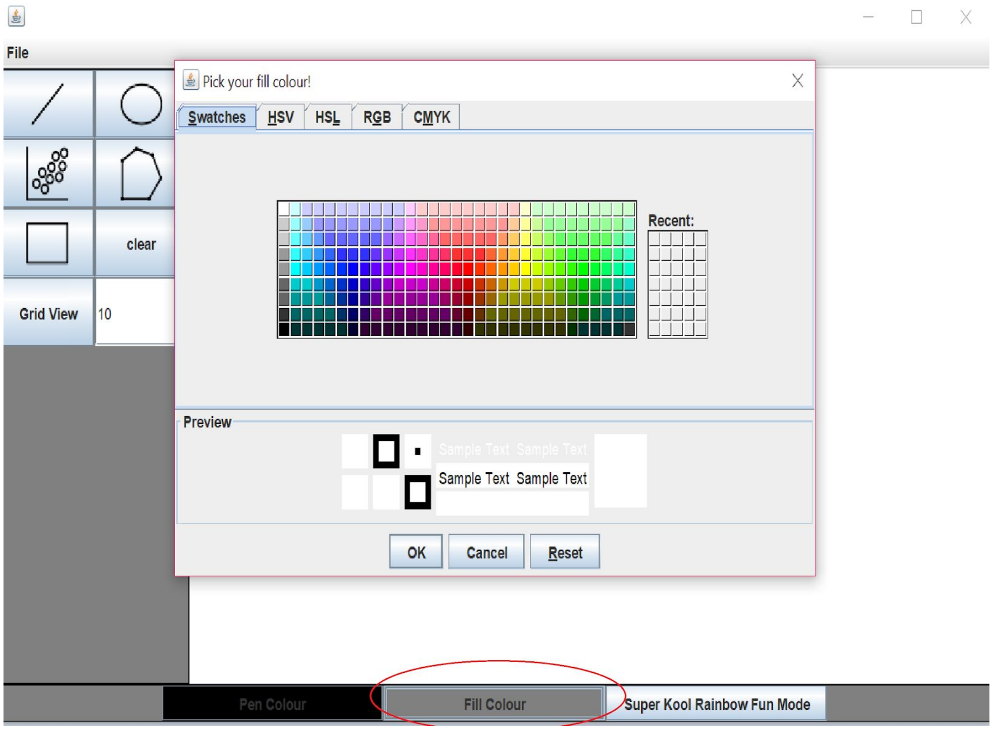
Change Pen Colour



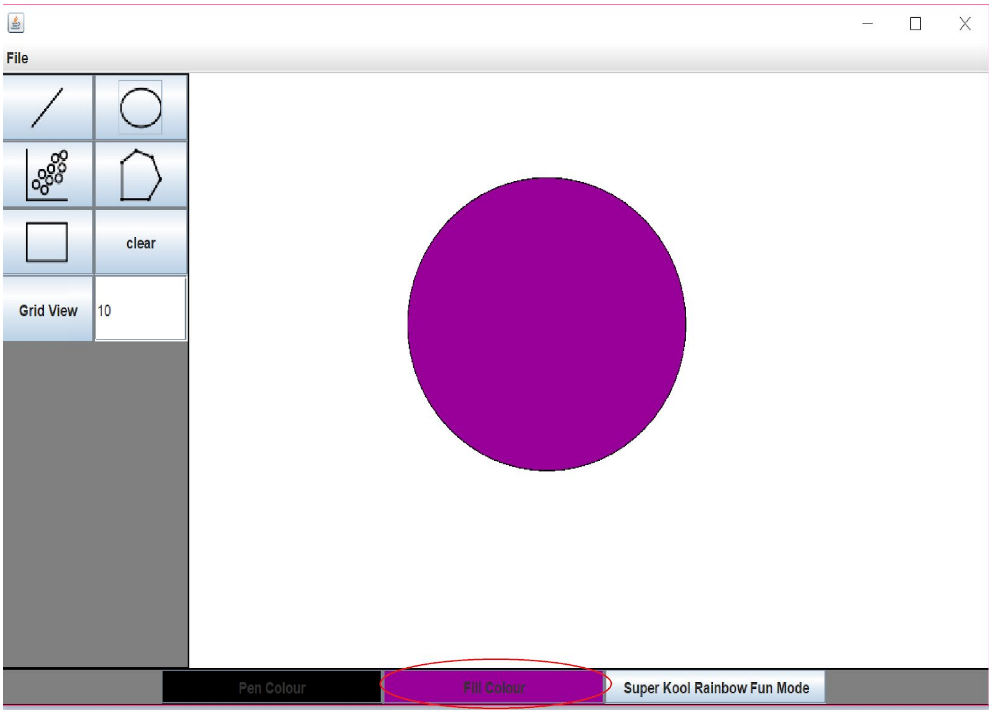
To change the pen colour, select the button circled in red. This will open up a few different options for selecting your pen colour. Cycle through the tabs, select your colour and press okay. You can now draw a shape with that pen colour as seen below.



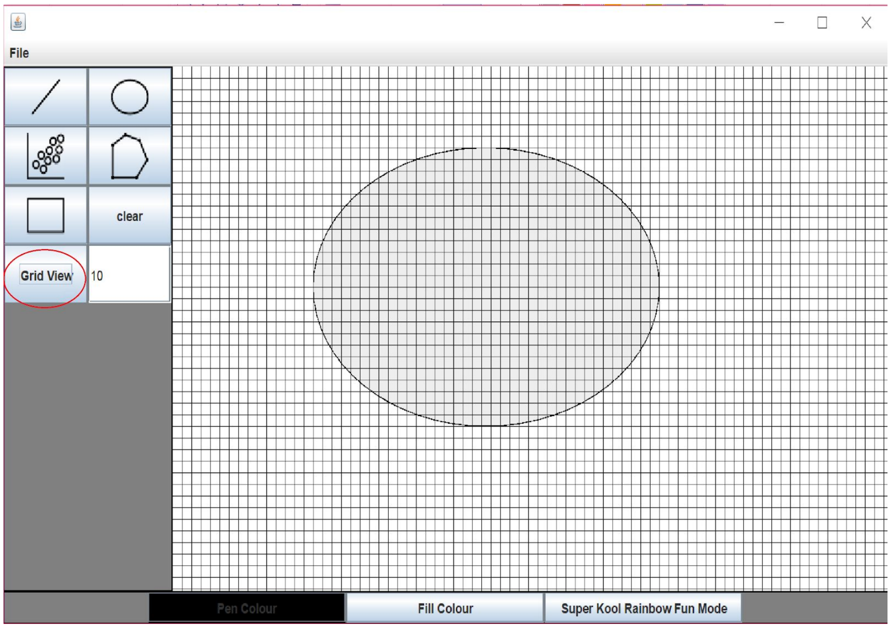
Change the Fill Colour



This works exactly the same as above, except you're selecting the fill colour button. Once a colour is selected you will get the below result.

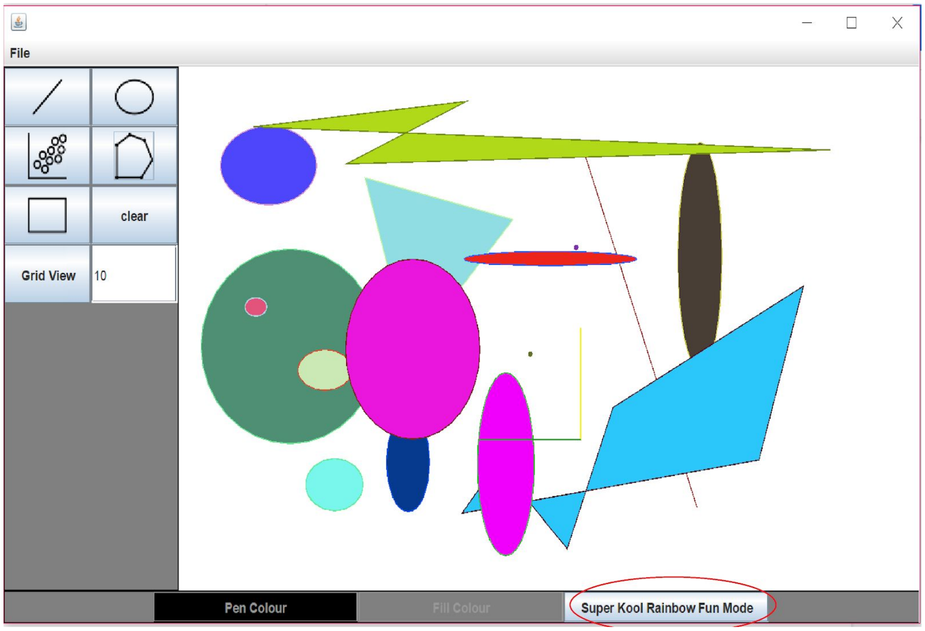


Grid View (Advanced Functionality)



By selecting grid view, the shape you draw will snap to a grid determined by the size entered in the text box next to the "Grid View" button.

Super Kool Rainbow Fun Mode (Advanced Functionality)



"Super Kool Rainbow Fun Mode" disables pen and fill colour options and randomly selects a hex colour value for pen and fill. As you can see above, each shape has distinct different pen and fill colours, randomly generated every time the user goes to make a new shape.