***Project Name***

***Drawing Equations***

***Names & sections:***

الاسم :هدى خالد يحيى غريب section 14

section 12 الاسم : منة الله هشام نور الدين

***Roles of each one:***

Menna : designing, implementing , and coding the 2 classes

( houghTransform & drawing ).

Hoda : designing, implementing , and coding the 2 classes

( houghLine & ProjectRun ).

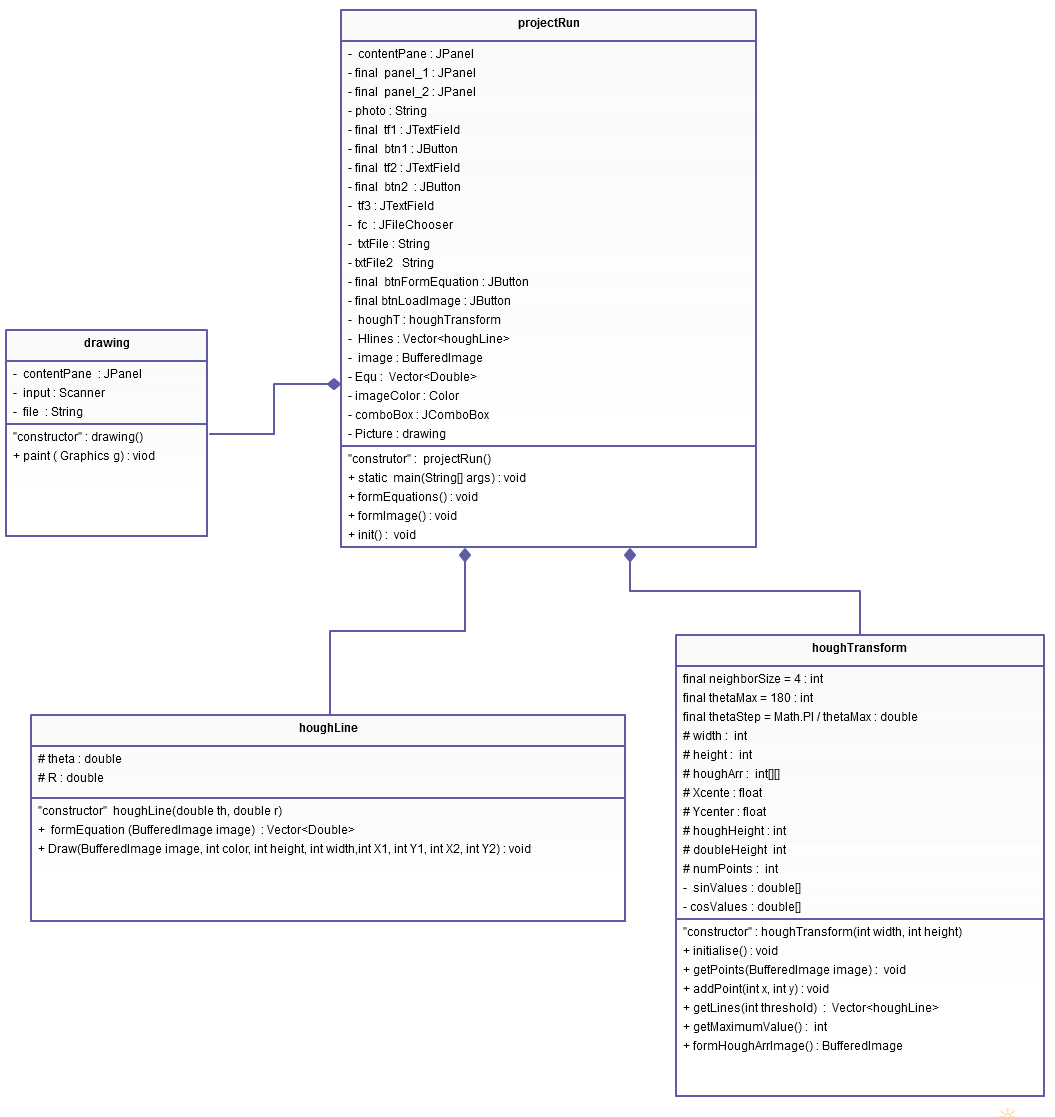
***Project description:***

The main idea of the project is to form equations for an image that you load through the program and after that you can get the image back from these equations by solving them.

The advantage here is that saving equations on a text file takes less memory on the drive than saving a whole image.

***Future work:***

Forming only one equation for the whole image by solving all the equations generated from the image.



***Code description:***

***Class houghLine :***

It forms an equation for each line generated from the hough transform.

This equation depends on an angle (theta) and a perpendicular line (R).

***Class fields:***

1. theta : value of the angle between the perpendicular on the line and the axis of the image
2. R : the length of the perpendicular from the origin to the line

***Class methods:***

1. **public** houghLine (**double** th, **double** r)

constructor which initiate the theta value ( angle between main axis in the image and the perpendicular on a line ) and R value ( perpendicular line ).

1. **public** Vector<Double> formEquation (BufferedImage image)

This function forms an equation for a line in the image and returns an equation for it along with its end points in the vector.

\* **@param** image: the image chosen by the user to form an equation for it

\* **@return** vector: a vector of double which contains the theta (which is the angle between perpendicular on the line and axis) and R (the perpendicular) along with the x-coordinates and y-coordinates boundaries of the line.

1. **public** **void** Draw(BufferedImage image, **int** color, **int** height, **int** width, **int** X1, **int** Y1, **int** X1, **int** Y2)

This function draws a line using the theta and R which is the perpendicular from origin to the line

\* **@param** image: buffered image to draw a line from an equation using theta and R

\* **@param** color: color chosen by the user to draw his new image with it

\* **@param** height: the height of the image

\* **@param** width: the width of the image

\* boundaries of the line:

\* **@param** x1

\* **@param** y1

\* **@param** x2

\* **@param** y2

//========================================================================//

***Class houghTransform:***

This is the class which forms the hough transform which is a method for detecting lines in an image along with canny's line detection.

***Class fields:***

1. neighborSize: The size of the neighborhood in which to search for other local maxima.
2. thetaMax: Number of values of theta to be checked.
3. thetaStep: Using thetaMax to get the step.
4. width, height: the width and height of the image.
5. houghArr: The hough array.
6. Xcenter, Ycenter: the coordinates of the center of the image.
7. houghHeight: the height of the hough array.
8. doubleHeight: double the hough height ( for negative values )
9. numPoints: the number of points that have been added.
10. sinValues, cosValues: Values of sin and cos for different theta values.

***Class methods:***

1. **public** houghTransform(**int** width, **int** height):

constructor for the class which initialize the width and height with the dimensions of the image

\* **@param** width

\* **@param** height

1. **public** **void** initialise():

Initializes the hough array. Called by the constructor can be used to reset the transform in case you need to add another image but it has to be of the same dimension.

1. **public** **void** getPoints(BufferedImage image):

**@param** image:

The image which is scanned to count the pixels which are assumed to be edges and adds them as points.

The image has to be in grey scale mode.

The image has to be of the same dimensions as the one passed to the constructor.

1. **public** **void** addPoint(**int** x, **int** y):

adds a single point to the hough array

\* **@param** x

\* **@param** y

1. **public** Vector<houghLine> getLines(**int** threshold):

This method extracts the lines from the hough array and saves them in a vector of hough line objects.

\* **@param** threshold:

to determine the minimum number of intersections between

curves of hough array

\* **@return** vector of hough line objects

1. **public** **int** getMaximumValue():

**@return** the maximum value in the hough array to be used in drawing hough

array image

1. **public** BufferedImage formHoughArrImage():

Gets an image for the hough array in the form of a buffered image with black color for the curves of hough array and white color for the back ground just in case you want to.

\* **@return** image of an equation which is precalculated

//================================================================//

***Class drawing :***

This class generates the drawing from an equation that's already made and saved on a file it uses graphics to draw it.

It extends the JDialog Class.

***Class fields:***

1. contentPane: JPanel object on which the drawing will take place.
2. input: input stream of type Scanner for opening a file and getting the equation.
3. file: String containing file name to be opened with the input stream.

***Class methods:***

1. **public** drawing( String FileName, Color color)

public constructor which initializes the color of the foreground ( drawing)

\* and the file name which contains the equation

\* it also sets the size of the frame on which the drawing will appear

\* **@param** FileName: the name of file to be opened

\* **@param** color: color of the foreground

1. **public** **void** paint ( Graphics g)

This function loads the equation of each line from the file and draws it.

It uses only its boundaries value for x and y as it draws line segments.

//====================================================================//

***Class projectRun :***

This class controls the interface of the project it also has objects from the rest of the classes created in this project.

It extends the JFrame class.

***Class fields:***

1. panel\_1, panel\_2: tabbedPane Panel
2. contentPane: main panel in the main frame
3. tf1, tf2, tf3: text fields used in the tabbedPane while browsing for files
4. btn1, btn2, btnFormEquation, btnLoadImage: Buttons used in tabbedPane for browsing generating equations and images
5. fc: JFileChooser used to choose your an image from your PC to form an equation for it in a file which you chooses too and to load image from a file you chooses too.
6. txtFile, txtFile2, photo: Strings Carry the extension of the images and files you choose in the project.
7. comboBox: JComboBox let you choose the color of the image.
8. Hlines: Vector carry the lines detected in the image you chooses to form an equation for it.
9. Image: Bufferedimage carry the image you chooses.
10. Equ: Vector carry the parameters of the equations calculated for the image.
11. imageColor: carry the color type you chooses from the JComboBox for your drawing.
12. houghT: an object from class hough transform which analysis your image and detect main lines.
13. Picture: an object of class drawing which generate a picture from an equation which is already made and saved

***Class methods:***

1. **public** **static** **void** main(String[] args)

The main function from which the project starts running.

1. **public** projectRun()

Constructor calls init function.

It initiates the value of the Strings containing files and image names.

1. **public** **void** formEquations()

Form an equation for every line in an image and saves it on a file.

1. **public** **void** formImage()

Converts an equation to an image by changing every equation saved on a

file for a line to a line and drawing it.

1. **public** **void** init()

Organizing and forming the user interface along with controlling the formulation of equations for images and converting the equations back to images.