CSCI-4448 Project Part 2

Team: Jonathan Vu

            Brandon Ruffin

            Robert Allen

Title: Paint

Project Summary:

Paint is a vector drawing application that supports drawing shapes such as lines, polygons, and ellipses. As a vector based drawing application, Paint will represent all shapes in an image as a set of coordinates and directions. Paint will attempt to create and save vector images in the common SVG format to allow cross compatibility with other vector drawing applications.

Requirements:

User Requirements

ID	Description	Priority
US-01	As a user I can save my image.	High
US-02	As a user I can load an image.	High
US-03	The user should be able to rotate the canvas.	Medium
US-04	The user should be able to select a brush size and color for all drawing actions.	High
US-05	The user should be able to draw straight lines.	Critical
US-06	The user should be able to draw polygons.	Critical
US-07	The user should be able to draw ellipses.	Critical
US-08	The user should be able to create textboxes.	Low
US-09	The user should be able to draw paths.	High
US-10	The user should be able to fill enclosed areas with a color. Enclosed areas being and 2D shape completely enclosed by a line.	Medium
US-11	User should be able to select and rotate or resize previously drawn shapes without distortion.	Nice-to-have
US-12	The user should be able to draw using tablet pen input.	Nice-to-have
US-13	The user should be able to resize the canvas.	Medium
US-14	The user should be able to crop the canvas.	Medium

Functional Requirements

ID	Description	Priority
FR-01	When selecting a color, the user should be presented with a color palette.	High
FR-02	When selecting brush size, the user should be presented with a list of numbers beside sample brush lines.	High
FR-03	All drawn shapes will be stored as vectors rather than bitmaps.	Critical
FR-04	Saved images will be stored on the local filesystem.	Critical
FR-05	The program should add visible anchor points to drawn shapes that can be used to modify the shapes.	Medium

Non-Functional Requirements

ID	Description	Priority
NFR-01	For usability, the images should be saved in the standard SVG format to allow cross-compatibility with other vector image editors.	Nice-to-have

Business Requirements

There are no business requirements for this project.

Use Cases:

UI Mockups:

Save Image
Load Image

Canvas ▶ Resize
Rotate

| File | Edit | | X |

Brush Size 12px ⬍ Tool Options

Multi-line
textarea
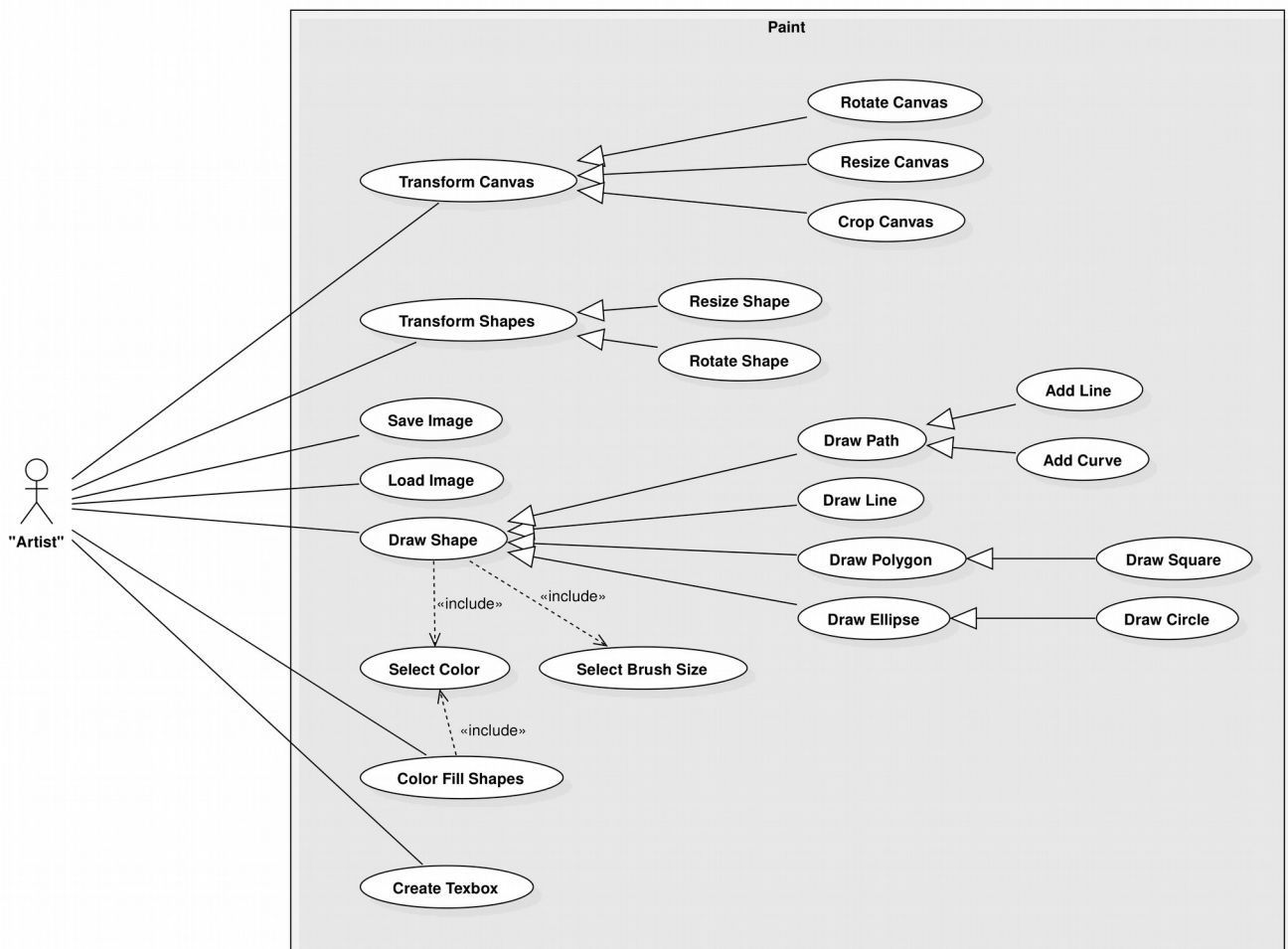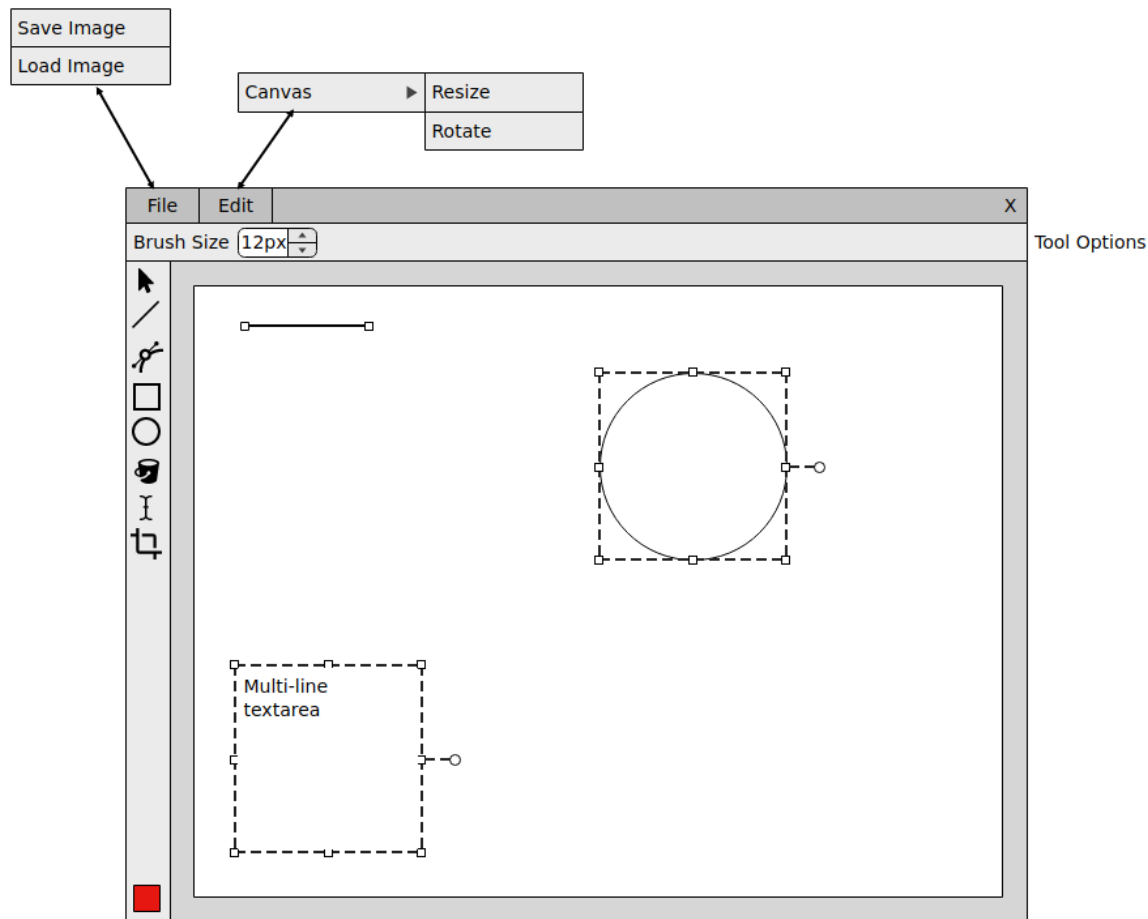
Save Image   X

| File Path | ... |

Save   Cancel

Load Image

| File Path | ... |

Load   Cancel

Select Color

R
G
B
H
S
B

#

OK   Cancel

Resize Canvas

Current:
     Width: 0px   Height: 0px

New:
     Width: [ ] px   Height: [ ] px
     Width: [ ] %   Height: [ ] %

Apply   OK   Cancel

Rotate Canvas

◉ Clockwise   ○ Counter clockwise

◉ 90°
○ 180°
○ 270°

Apply   OK   Cancel

Data Storage:

We are storing our data in the SVG file format. We intend to persist any and all objects drawn to the canvas and their properties such as their color, position, width, and height.

Class Diagram:

SaveDialog
-savePath: JTextField
-saveButton: JButton
-cancelButton: JButton

-isValidPath(path: String): bool
-save(path: String): void

LoadDialog
-loadPath: JTextField
-loadButton: JButton
-cancelButton: JButton

-isValidPath(path: String): bool
-load(path: String): void

Paint
-window: JFrame

ResizeDialog
-newWidth: JTextField
-newHeight: JTextField
-okButton: JButton
-cancelButton: JButton

-resize(width: int, height: int): void

JButton
+command: AbstractAction

AbstractAction
+name

+actionPerformed(): void

SVGFile
+File: file
+xmlVersion: String
+xmlEncoding: String

+save()
+load()

Canvas
-width: float
-height: float
+uiButtons: JButton[1..*]

+drawObjects(): void
+rotate(angle: float): void
+resize(newWidth: int, newHeight: int): void
+crop(x1: int, y1: int, x2: int, y2: int): void

RotateDialog
-isClockWise: bool
-angle: JTextField
-clockWise: JRadioButton
-counterClockWise: JRadioButton
-okButton: JButton
-cancelButton: JButton

-rotate(cw: bool, angle: float): void

«interface» XML
+toXml(): String

«interface» Drawable
+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void

Shape
+id: String

+rotate(angle: float): void
+fill(rgbValue: String): void

Style
+fill: String
+fillRule: String
+stroke: String
+strokeWidth: String
+strokeLineCap: String
+strokeLineJoin: String
+strokeOpacity: String

Rectangle
+width: float
+height: float
+startPoint: Point

+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void
+toXml(): String

Ellipse
+rx: float
+ry: float
+startPoint: Point

+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void
+toXml(): String

PolyLine
+points: Point[1..*]

+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void
+toXml(): String

Text
+text: String
+startPoint: Point

+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void
+toXml(): String

Path

Point
+x: float
+y: float

Line
+startPoint: Point
+endPoint: Point

+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void
+toXml(): String

BezierCurve
+startPoint: Point
+startControlPoint: Point
+endPoint: Point
+endControlPoint: Point

+draw(canvas: Canvas): void
+undraw(canvas: Canvas): void
+toXml(): String