Project 4

"Pain" Brush

Objectives

The objectives of the project are twofold

1) To implement the Object Oriented Programming concepts such as Inheritance and Polymorphism.

2) To introduce students to a graphics library and learn how to write and event driven program.

SECTION 1

Application and Features

The program is a basic drawing utility like the Paint Brush program on Windows. This program will allow the user to draw basic shapes such as triangles, rectangle, polygons circles, lines, etc. The user chooses a particular color from a palette to draw these shapes, and also can use a bucket-tool to fill up the interiors of closed shapes, an eraser tool to delete a shape, and save and load from a file on the disk.

- The drawing mechanism will work like this:

  - The user will first choose the type of shape he wants to draw from a panel of available shapes.
  - The user will then click the points, that is, the vertices of the shape, and then the program will complete the shape using these points. For example, if the user choose the triangle shape and then left-clicks the mouse on three different points on the canvas, those three points will be used to draw the triangle.
  - Once a shape has been drawn, it will be added to a list (an array of shape type pointers) of all shapes, called *allShapes*, where the objects of the various shapes drawn so far have been dynamically allocated and their pointers (of type shape) has been stored.
  - Note that the array allShapes itself needs to be dynamically allocated, as its size might vary as new shapes are added and deleted. At any point its size should be **no more than** twice the number of shapes stored in it.

- Shapes can be deleted like this:

    - The user will pick the “eraser” tool from the panel.
    - The user can then click anywhere on the canvas, and if the point of the click is inside a shape, that shape will be deleted from the allShapes array.
    - This will go on until the user “un-picks” the eraser tool.

- Shapes can be filled like this:
  Note: this tool will only change the boundary color in case of curves and lines.

    - The user will pick the “bucket” tool from the panel.
    - The user can then click anywhere on the canvas, and if the point of the click is inside a shape, that shape will be filled with the current color.
    - This will go on as until the user un-picks the bucket tool.

- The colors: the program will allow (at least) Red, Green, Blue, Yellow, Orange, White and Black

- The file can be saved like this:

    - At any point during drawing the user can select the save option from the panel.
    - The program will then save all the shapes in the allShapes array in a file called “drawing.pb”.

- A file can be loaded into the program at the start:

    - When the application starts it will ask the user to either load the saved drawing or start a new one.
    - If he chooses to start a new drawing the older one will be overwritten, otherwise, the contents of drawing.pb will be loaded into the allShapes array and the user will proceed drawing from there.

- What will the panel look like:

    - The panel is supposed to be a vertical band on the left side of the canvas, with square shaped “buttons”, by clicking which the user can select various options.
    - There are buttons for: shapes, colors, eraser tool, bucket tool and save option.

SECTION 2

Classes and the Hierarchy

The array **allShapes** contains Shape type pointers to various objects that have been drawn so far. All these objects are of classes belonging to the Shape hierarchy, with an abstract class called Shape on the top. Here's a point-wise description of this hierarchy.

- In the following "**Point**" is a struct containing "doubles" x and y, for the x and y coordinates of a point on the canvas.

The Classes

- What are the various shapes allowed by the application:

    - At minimum the program allows the user to draw the following (the number of Points the shapes contains are given in the brackets)

    - **Line** (2: the endpoints), available is two styles: dotted and solid.
    - **Triangle** (3: the three corners)
    - **Rectangle** (2: two opposite corner)
    - **Quadrilateral** (4: the four vertices)
    - **Circle** (1: the center, although the user will have to specify the radius by clicking a second time on any point on the required circumference)
    - **Polygon** (n: any number of points the user wants to specify, he will eventually click the right mouse button to indicated that he has given all the points)
    - **Curve** (**n:** any number of points the user wants to specify, he will eventually click the right mouse button to indicate that he has given all the points.)
        - **Note:** for us a curve is drawn similarly to a polygon (by joining points with straight lines, but it is not closed like a polygon: first and last points are not joined)

The Hierarchy

- You should infer the Hierarchy from the following statements:

    - A Shape is either an OpenShape, a Circle, or a Polygon

- An OpenShape is either a line or a curve.
- A Polygon: could be a general Polygon (with n points), a Quadrilateral (with four points), or a Rectangle (with just two points)

- Attributes common to all Shapes
  - A list (array) of Points that are vertices of that shape.
  - Integer, called color: indicating the boundary color
  - const Integer, n: number of points

- Attribute common to all Polygons

  - Integer, fillColor: indicates the color with which the polygon is filled. If it is not filled this attribute contains some exceptional value.

- Attribute special to a Circle:
  - Double, radius: radius of the circle

- Attribute common to all OpenShapes:
  - Boolean, style: true if the line is solid, false otherwise.

- Methods common to all Shapes
  - **void draw():** draws the shape on the canvas, and immediately stores the new shape in the allShapes array.
  - **contains(Point p)**: returns true is the shape contains Point p either in the interior or on the boundary. Lines and curves will also implement this function and return true if p lies anywhere "on" them.
    - To implement this, read about the **Ray Casting Algorithm**: which is a very simple way to detect if a point is inside or outside a closed shape.
    - In case of lines and curves you simply need to detect if the point is on the edges making them up.
    - You will find it useful to implement a utility function called pointLiesOnLine(Point a, Point b, Point c), where the first two points are endpoints of a line and the third point is to be tested: if it lies on the line you return true, else false.

- Methods common to all Polygons, and the Circle
  - **void fill(int color)**: fills the closed shape with the given color
    - Repeated use of contains function will help you do this.

- Method common to all OpenShapes

- **void changeColor(int color)**: changes the boundary color.

SECTION 3

The Graphics Library

You need to include the file provided with this statement.

SECTION 4

Additional Points on Coding

- While loading from the file the program will allocate all shape objects for the saved shapes and put them in the allShapes array. Then it will go throw the array and draw (and fill in required) each shape on the canvas.
- There should be no memory leaks anywhere in the program, at any time during execution.
- When the eraser tool is used to delete a shape the pointers will have to be shifted back by one place each, and if the number of remaining shapes becomes less than half the size of the allShapes array then the size of the array should be halved. This will allow us to maintain the condition that the size of allShapes should never be more than double the number do shapes it actually stores.

That’s all!