

Design of Sketch Pad Drawing Tool

Report

By,

Hareem Nisar - 250947339

Rajaraman Ganesan – 251056279

Sai Deepthi Rajaputra - 251002004

Introduction:

This Sketchpad tool makes human and a machine to interact easily through diagrams. Heretofore, most **interaction** between men and machines slowed down by the need to reduce all communication to written statements that typed; in the past, we have been writing letters to rather than conferring with our computers. The Sketchpad system, by eliminating printed comments in favour of line drawings, opens up a new area of man-machine communication.

The decision to implement a drawing system did not mean, however, that **brute force techniques** were to be used to computerize ordinary drafting tools; it was implicit in the research nature of the work that simple new facilities should discover which, when implemented, should be useful in a wide range of applications, preferably including some unforeseen ones. It has turned out that the properties of a computer drawing are entirely different from those of a paper drawing not only because of the accuracy, ease of illustration, and speed of erasing provided by the computer.

Sketch Pad & the Design Process:

Construction of a drawing with Sketchpad is itself a model of the design process. The locations of the points and lines of the drawing model the variables of design; and the geometric constraints applied to the ends and edges of the drawing model the design constraints which limit the values of design variables. If such design conditions added to Sketchpad's vocabulary of constraints, the computer could assist a user not only in arriving at a lovely drawing but also in arriving at sound design.

Shapes & other Functionalities:

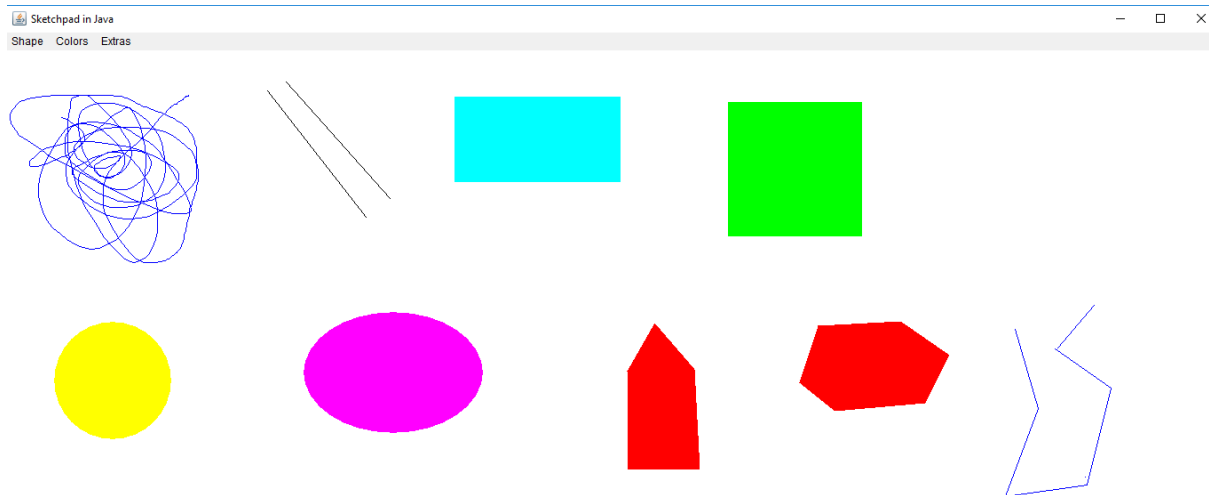
The sketchpad tool consists of a design of flowing shapes:

- i. **Freehand:** A drawing form did only using eye and hand coordination. In other words, a type drawing without any ruler or any shape materials.
First, we initialize the x and y coordinates, height, width and colour of the object to be drawn. To make a freehand sketch, we map the points to x and y using **stream API** used to process the collection of objects. To detect the line drawn, we make use of **for each loop** used instead of the indexed array element and also used to iterate over an array.
- ii. **Line:** Series of points which extends in both sides It is uni-directional.
Here tends the same as initiating x, y, height, the width of the object to be drawn. To create points we use mathematical line equation $(x1 + x2) / 2$, $(y1 + y2) / 2$.
- iii. **Square:** Set of four lines whose length is equal to breath.
For designing square height, width, co-ordinates and Type of object initiated. A square has all sides similar to make the program, as height equals width to get the correct result.
- iv. **Rectangle:** Simple plane whose length is more than its breath.
Here the same operation as for square is performed, but the difference is we don't make functionality like height equals width.
- v. **Circle:** A diagrammatic representation of points in round shape with equal circumference.
To draw a circle, the initialization is done and to confirm whether the mouse hovers over the object, we make a condition like a panel coordinate must be more excellent than object coordinate to satisfy the requirement.
- vi. **Ellipse:** Oval shape points moving along the line of circumference.
Ellipse has a similar function as a circle but the shape drawn will slightly vary and in the case for square and rectangle

vii. Polygon:

- a. **Open Polygon:** A polygon is free when the segments do not all connect at the beginning and end. That is, if we draw the polygon starting at one point, we finish drawing at a different point
- b. **Closed Polygon:** A polygon closed when the segments do connect at the beginning and end. That is, you start to draw the line at a location and finish at the same position.

The Shapes designed in sketchpad displayed below:



Main Functionalities:

The Functions enabled for the Simple Sketch Pad Tool listed below:

- i. **Move Object:** Selecting on the co-ordinate to move a particular object from current location to new location. To move the selected object, we hover the mouse over the object, when we click on it click event comes into place. We use **action. Move** to move the selected object from its current location to a new position.
- ii. **Move Group:** Select the objects to move and pull it to the desired location. Here we move all the object on top of each object perform click operation on top of co-ordinate and move the bunch of purpose to a new place we need.
- iii. **Cut:** On selecting this, functionality and clicking, the object makes the object to disappear from the current location. To reduce the particular purpose, we click on it. If it finds object coordinate, it will cut the object, or else it returns null.
- iv. **Paste:** This used to make the disappeared object to appear in the new location whichever we need. Similar action for paste operation it clicks on the empty pane to paste the recently deleted object. The paste operation done only once.
- v. **Undo:** A feature that allows a user to cancel or reverse the last one or more commands executed. To undo the objects drawn, we create an **empty stack**. Therefore, whenever we perform the undo option, the object placed inside the stack.
- vi. **Redo:** The redo function restores any actions that have been previously undone using an undo. This is vice versa operation of undo operation. When redo pressed, the object **released from the stack** orderly.
- vii. **Clear Canvas:** This will erase the entire currently present object and make the pad to look freshly. Each time we clear the working screen, the new array list created for storing new input objects.

- viii. **Save:** Used to save our work for future purpose. To keep the created diagram new buffer created. On clicking save option, a new pop up window appears where we enter the filename saved, and it stores the image in **png format**.
- ix. **Load:** To check the saved work in case we need to refer in future — same operation for loading the saved image. When we click on load, the small pane appears where we give the saved file name, and it opens and shows our saved progress.

The Design of Simple Sketch Pad done using Java AWT. The key terms used during the programming phase listed below:

The Libraries used given below:

- a. **AWT [Abstract Window Toolkit]:** Platform dependent API for creating Graphical User Interface for java programs. Here components displayed according to the view of the operating system.
- b. **ArrayList:** It provides us with dynamic arrays in Java. Inherits `AbstractList` class and implements `List` interface. Java `ArrayList` allows us to access the list randomly.
- c. **List:** It is an ordered collection of objects in which duplicate values can store list Interface implemented by `ArrayList`, `LinkedList`, `Vector` and `Stack` classes.
- d. **Collectors:** `Collectors` is a final class that extends `Object` class. It provides reduction operations, such as accumulating elements into collections, summarizing aspects according to various criteria.
- e. **ImageIO:** `ImageIO` is a utility class, which provides lots of utility method related to images processing in Java. Most common of them is reading from an image file and writing images to file in java.

The Event listener represents the interfaces responsible for handling events.

The Listeners used are:

- a. **Action Listener:** The listener interface for receiving action events. The class that is interested in processing an action event implements this interface and the object created with that class registered with a component. For instance, when the user clicks a button, chooses a menu item, presses Enter in a text field. The result is that an **actionPerformed** message sent to all action listeners that registered on the relevant component.
- b. **Window Listener:** The listener interface for receiving window events. The class that is interested in processing a window event either implements this or extends the abstract `WindowAdapter` class. The listener object created from that class then registered with a `Window` using the window's **addWindowListener** method. When the window's status changes by an action like opened, closed, activated or deactivated, iconified or deiconified, the appropriate method in the listener object invoked, and the `WindowEvent` passed to it.
- c. **Mouse Listener:** The listener interface for receiving, mouse events such as press, release, click, enter, and exit on a component. A mouse event also generated when the mouse cursor enters or leaves apart. When a mouse event occurs, the appropriate method in the listener object invoked, and the `MouseEvent` passed to it.
- d. **Mouse Motion Listener:** The listener interface for receiving mouse motion events on a component. The listener object created from that class then registered with a part using the component's **addMouseMotionListener** method. A mouse motion event generated when the mouse moved or dragged. When a mouse motion event occurs, the appropriate method in the listener object invoked, and the `MouseEvent` passed to it.
- e. **Item Listener:** The listener interface for receiving item events. The class that is interested in processing an item event implements this interface. The object created with that class then registered with a component using the component's **addItemListener** method. When an item-selection event occurs, the listener object's `itemStateChanged` method invoked.

By using the event listeners and other criteria, the design of sketchpad is programmed and executed well using **JAVA AWT** with the desired output.

State Chart:

A Statechart diagram describes a state machine. State machine defined as a machine, which encompasses different states of an object, and these states controlled by external or internal events. Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime, and these states changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems defined as a system that responds to external or internal events.

Statechart diagram describes the flow of control from one state to another state. States defined as a condition in which an object exists, and it changed when some event triggered. The most crucial purpose of a Statechart diagram is to model the lifetime of an object from creation to termination.

Here in the design of sketchpad all the functionalities like drawing shapes, colours, move, draw, move group, cut, paste, undo, redo, save and load get action from primary object drawing tool. If the action command given is equivalent to the function present, then action will be performed else it will return null operation. The pasting of the object will take place only once, and next time the clipboard will display null("Nothing to Display"). Here the class Drawing tool has a significant role to function the object drawn in various states.

Class Diagram:

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects. It is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing an executable code of the software application. It is also known as a structural diagram.

The purpose of the class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. Class diagrams have a lot of properties to consider while drawing, but here the diagram will be regarded as from a top-level view — the class diagram considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system, but they are also used to construct the executable code for forward and reverse engineering of any order.

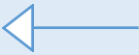



Object Diagram:

The object is an instance of a particular moment in runtime, including objectives and data values. A static UML object diagram is an instance of a class diagram; it shows a snapshot of the individual state of a system at a point in time. Thus an object diagram encompasses objects and their relationships at a point in time. It considered a particular case of a class diagram or a communication diagram.

During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then generate a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram. Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate particular examples of the classifiers that are required.

Object diagrams submitted with the project based on the created of different shapes. The first object diagram describes the layout of the sketchpad. All the following diagrams show the user draws a relationship between different classes (e.g. Shape, Colour, Graphics) as each shape (e.g. blue line or red polygon). Essential attributes and relationships that differentiate between class highlighted via font-colouring in the diagram. Note that instead of building each shape from scratch, we are using **JAVA built-in 'Point' and 'Graphics'** classes. We have created seven classes (Line, Freehand,

Circle, Ellipse, Square, Rectangle and Polygon), all extending from a class called '**Shape.**' The relationships used in the diagram and their brief explanation given below:

	Arrow type	General example	In Sketchpad example
Inheritance		B extends A	Ellipse extends Shape
Dependence		The function of B takes A as an input argument.	Function 'draw' of Circle takes Graphics as an input.
Aggregation		To Create B, we must have A. A is an input parameter for B's constructor	Type is input to Shape's constructor
Composition		B has A. A instantiated within B	Point instantiated within Line

Conclusion:

Thus, we have seen a basic understanding of what is sketchpad tool is, what all the shapes colours included, and what are all the functionalities done how it is implemented using JAVA AWT and displayed. We have also learned about what is object diagram & state chart its purpose of using it.