University of St Andrews | FOUNDED 1413

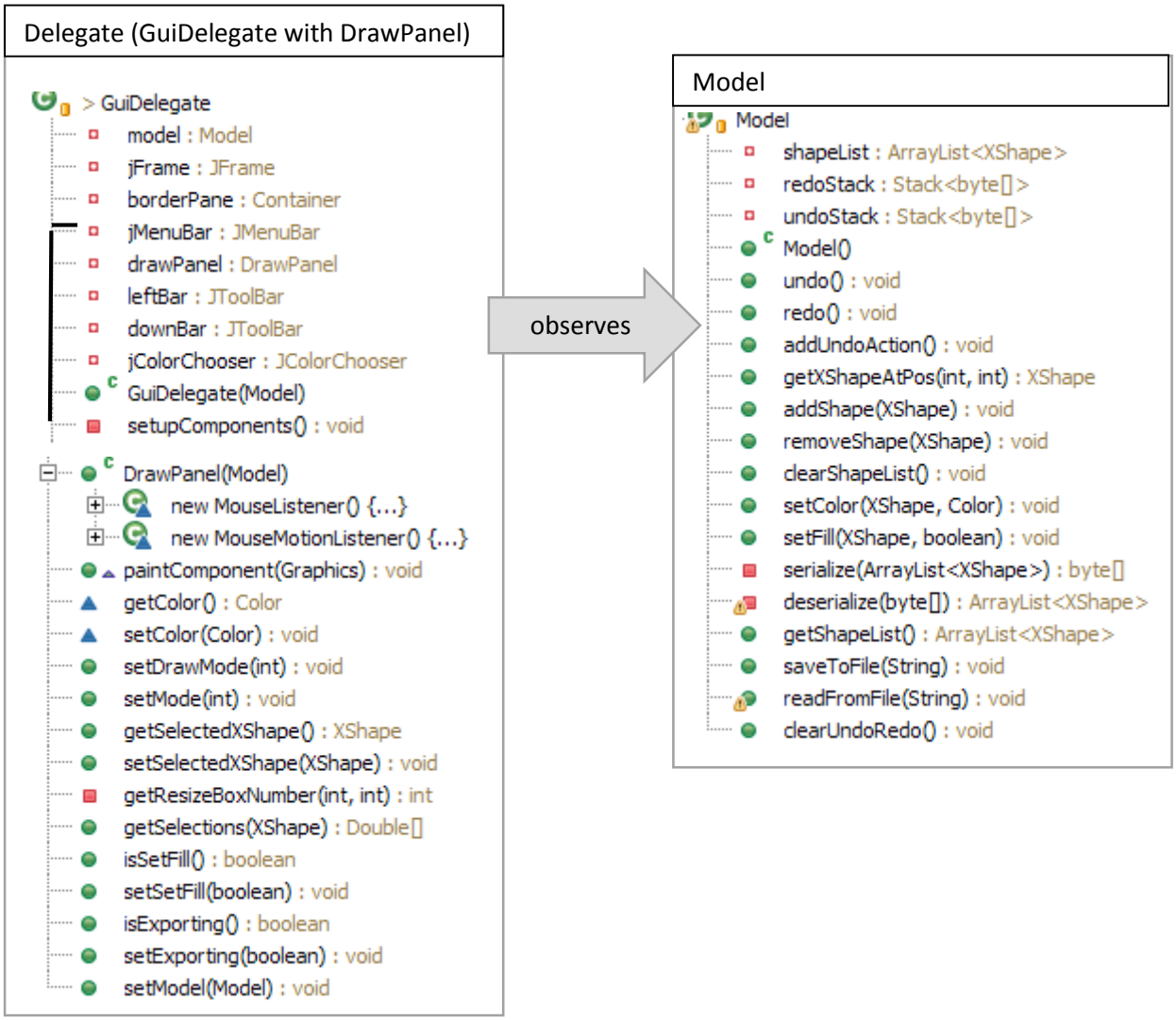
CS5001 Practical 4
Vector Drawing Package
Report
#160022011

Table of Contents

System Design: Design Decisions

The developed system uses a Model-Delegate (MD) design pattern. The MD pattern is a simplification of the Model-View-Controller pattern as it merges Controller and View into a Delegate. The MD pattern was chosen as the GUI is not extremely complex and therefore does not need to separate Views and Controllers. This means that the Graphical User Interface (GUI) including its controlling functions are implemented in a GuiDelegate class. Moreover, MD was used, as this approach improves the way of dealing with user inputs and outputs by separating them from the business logic, which is happening within the Model.

In order to link the Model with the Delegate even though the two elements are separated, the Observer-Observable pattern was used. As demanded by MD good practice, the Model is not able to see the GUI but the Delegate observes the Model and reacts accordingly to changes within the Model. If something changes within the model (e.g. a shape is added, removed etc.), the GuiDelegate is notified and consequently updates itself and its drawing panel (implemented as a JPanel) (e.g. shapes are redrawn etc.). To be able to observe the Model, the GuiDelegate holds an instance of the Model. It also calls the model's methods when events happen within the GUI (such as button presses etc.).



As the Model is responsible for the business logic it implements a list of shapes and provides methods for example to add, delete or select shapes, which can be called by the Delegate after a certain event (such as mouse clicks) has happened. Also the Model implements undo and redo functionality. Undo/redo works with two stacks that hold undoable and respectively redoable instances of the shape list in a serialised state. Just before a shape is for example added to or removed from the list, the Model pushes the current state of the list on the undo stack. If an action is undone, the last element of the stack is set as current state. Before the action is undone the undo method pushes the current state onto the redo stack, so it can be redone again. If a new action is performed, the redo stack is emptied so that "obsolete" redoable versions do not interfere and "undo" new changes again. The Model is also responsible for loading the shape list from and saving it to file.

In order to write the program in a re-usable and organized manner the model holds a list of objects of the abstract superclass XShape (Extended Shape). Subclasses of XShape, such as XEllipse or XHexagon inherit from XShape and implement its inherent methods such as construct, drag, resize and paint. An XImage inherits from XRect, as every Image is nothing else than a rectangle with image content inside. By using abstraction for the different XShapes, the Model and the Delegate have the possibility to use the shapes' methods consistently. In addition to the actual shape every XShape holds further drawing information such as colour and fill. This way the GUI can get this information from every shape in a consistent way and draw the shapes accordingly.

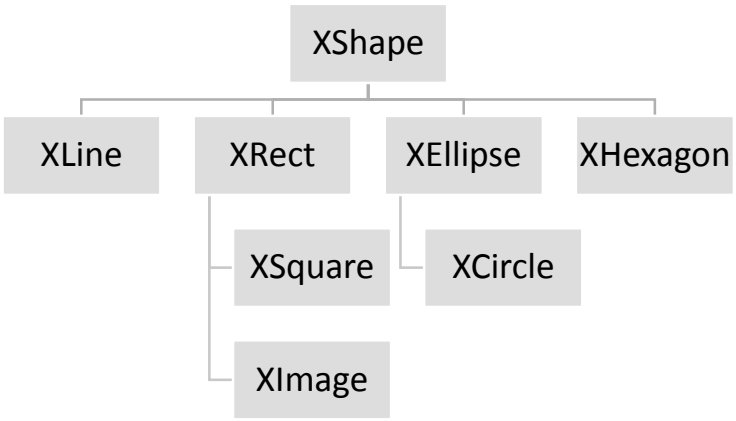


Figure 1: Hierarchy of the shapes

How the system works

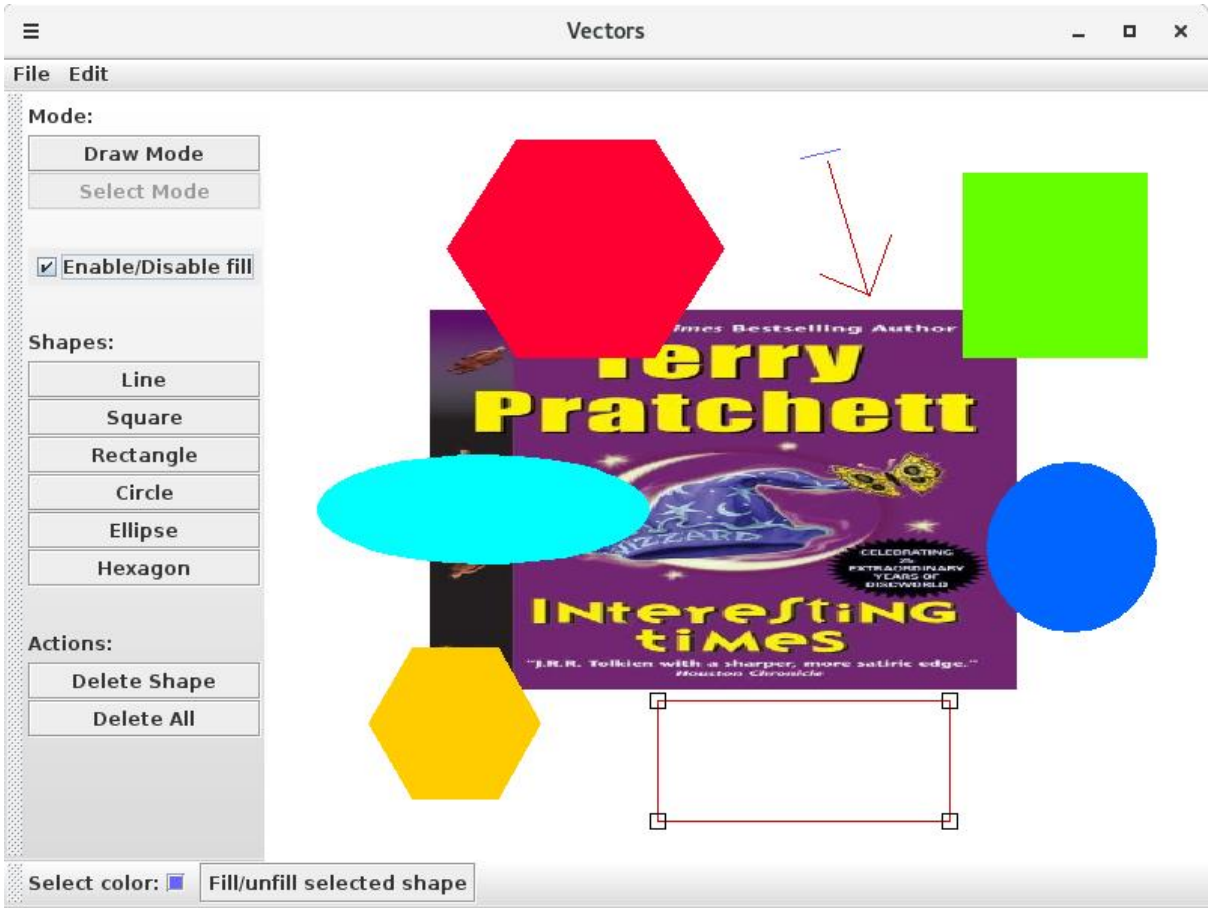
The system has two different modes – a draw mode and a select mode, which can be toggled with two particular buttons. If the draw mode is enabled, the specified shapes (Lines, Ellipses & Circles, Rectangles & Squares and Hexagons) can be drawn. The drawing process starts with the mouse press and stops as soon as mouse dragging has stopped. The drawn shape is then added to the shape list. Before drawing, it can be chosen which colour the shape shall have and whether it will be filled when drawn.

If the select mode is enabled, selected shapes can be dragged, resized, recoloured, re- or unfilled and deleted. A shape is selected when it is clicked on with the mouse. When the mouse click event takes place, the Delegate forwards the mouse position to the Model which searches the list of shapes, if any of the shapes contains the mouse coordinates. If so, the top shape that contains the coordinates is returned. If none of the shapes contains the coordinates, the Model returns null. When a shape is selected, the GUI highlights the selected shape by displaying four resize boxes, one on each corner of the shape. If one of those boxes is dragged, the shape gets resized. If the shape itself is dragged, it can be moved on the screen and finally stays at the position where it was dropped by the mouse. The system also allows to import images such as JPG, GIF, PNG which can then be moved and resized on the screen.

All actions (drawing, deleting, resizing, moving, colouring, filling & importing images) can be undone or redone again by clicking on `Edit > Undo step` or `Edit > Redo step`. Also clearing the complete panel by pushing the "`Delete All`" button is undo-/redoable. New projects can be started by clicking on `File > New project`. In contrast to the "`Delete all function`", previous steps are not redoable when starting a new project, as the stacks are cleared.

The project can be saved to file and loaded from file again so that the vector drawings can be worked on again after loading. To save a file, the system saves a serialized version of the shape list. When loading a file, the system de-serializes the file again and sets it as the current shape list.

Finally, the project can be exported in PNG format. This works with `File > Export to PNG`.



Problems

There are two minor problems that come with using the resize boxes. When lines are resized crosswise and then resized again, they change their orientation from top-right/bottom-left to top-left/bottom-right. This is because the resize does not check if the line is rising or falling for every case where a different resize box is used. This can be fixed by checking what orientation the line has and resize accordingly. The other problem is that circles and squares are not yet resizable crosswise as this function works with absolute values of the shortest side and therefore needs to take into account four different cases per used resize box. This can be fixed by implementing every case for every resize box.

Extensibility

The system implements all the basic and enhanced requirements (apart from orientation changes). Furthermore, following extensions were implemented:

- Shapes can be filled and unfilled
- Shapes can be deleted
- The panel can be cleared be deleting all shapes
- Undo and Redo buttons are deactivated when no undo/redo actions are possible
- A new project can be started
- The project can be exported to the PNG format

When further extending the application, it would be reasonable to extract the undo/redo functionality into its own class. Also the model can be designed in order to take more information into account, such as the size of the drawing panel and its background colour, which can be easily done as the application provides high extensibility through its design.