CMPT 381 Assignment 4
Ink, Grouping, and Clipboards

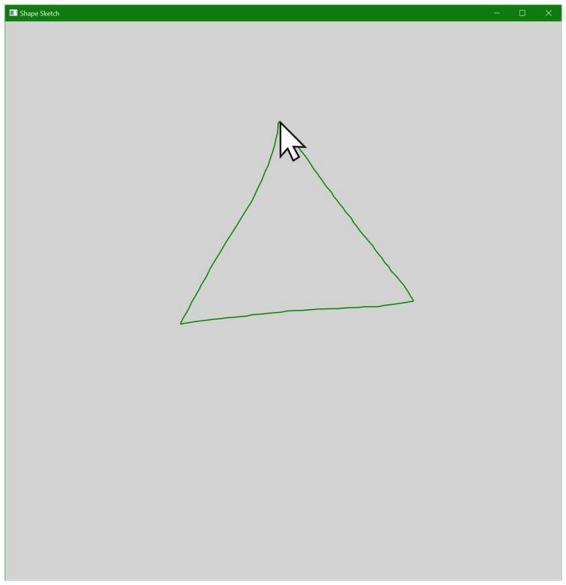
Due: Friday, December 4, 11:59pm

Overview

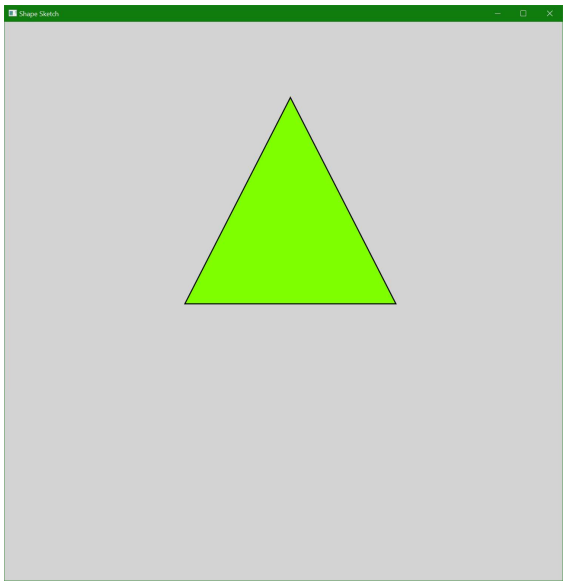
In this assignment you will build a JavaFX application for sketching shapes. The user draws the shapes freehand as digital ink, and the ink is converted to lines, rectangles, circles, and triangles if the ink is a close match to the shape. Users can select shapes and move them by dragging, and can rotate them using a single rotation handle at the centre of a selected shape. A selected set of shapes can be grouped, and then moved or rotated as a group. Selected shapes or groups can be cut, copied, and pasted using standard keyboard commands (Ctrl-X, Ctrl-C, and Ctrl-V).

Part 1: Digital ink and shapes

The main part of the application is a visual sketchpad where the user can draw freehand shapes using digital ink, and the system will recognize shapes and convert the ink to the appropriate shape. The user can draw straight lines, rectangles, circles, and triangles (an example of creating a shape is shown in the images below). Once a shape is created, the user can click on it to select it, and can move it (by dragging) or rotate it (by dragging the rotation handle shown at the centre of a selected shape).



The user has just finished drawing a triangle on the screen using digital ink

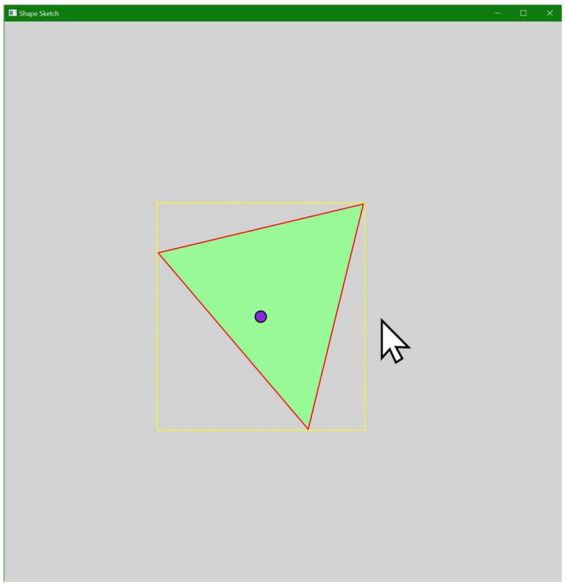


After releasing the mouse button, the system recognizes the ink as a triangle, and creates a triangle shape

Interaction Requirements:

- When the user presses the mouse button and drags, the system records and shows a line for the digital ink.
- When the user releases the mouse button, the system compares the ink to the four types of shape, and if the ink is a close match, the system creates an appropriate shape object. If the ink is not close enough to any shape, the ink disappears and nothing is created.
- If the user presses the mouse down on a shape (or in the case of a line, close to the line), the shape is selected and can be moved by dragging with the mouse.
- The system shows selected shapes by changing their outline colour to red, drawing their bounding box in a yellow dashed line, and showing a circular rotation handle at the centre of the shape (see picture below).
- Selection is persistent, so a selected shape stays selected until the user clicks on a different shape (which then becomes selected) or on the background (which cancels any selection).
- Shapes can be rotated by clicking on the rotation handle and dragging left or right; the amount of rotation and direction of rotation is controlled by the horizontal movement.

- Note that the bounding box is not rotated along with the shape (which means that the dimensions of the bounding box change as the shape rotates).
- If a shape is selected, pressing the Delete key on the keyboard removes the shape from the system.



The user has selected the shape, then clicked on the rotate handle and dragged to the right

Code Requirements:

- Use a standard MVC architecture, **but** use the hierarchical drawing capability described in lectures (i.e., where a shape is able to draw itself).
  - Create classes ShapeModel, ShapeView, ShapeController, and InteractionModel for the MVC components.
  - Create classes XLine, XRectangle, XCircle, and XTriangle to represent shapes in the system; you may wish to create abstract base class XShape to consolidate some of the standard code that is common to all shapes (optional)
  - Create class DigitalInk to store the points drawn by the user, and keep an instance of this class in InteractionModel.
  - ShapeView must use immediate-mode graphics.
  - When drawing circles, draw a cross on the circle so that rotation can be seen clearly
  - Use publish-subscribe communication between the model and the view.
  - ShapeView must initiate the drawing of the shapes on the screen (although the actual drawing can be delegated to the shapes themselves).
  - ShapeController must implement a state machine to handle user interactions.
  - Create class RotationHandle to keep track of the location, selection, and drawing of the rotation handle; an instance of this class should be in each shape.
    - The location of the rotation handle should be the centre of the bounding box
    - Because bounding boxes are adjusted during rotation, do not recalculate the location of the rotation handle during rotation; the location can be recalculated when the rotation finishes.
  - The InteractionModel should store the selection.
- Since we have only one view, you do **not** need to use normalized coordinates (using pixel coordinates may be useful for debugging in later parts of the assignment).
- To implement rotation and translation, do **not** use JavaFX transforms; instead, use the basic equations discussed in lectures that calculate new coordinates for each point that is used to draw the shape.
- Use the "compare to perimeter" technique described in lectures to determine whether the ink can be converted to a shape. (You may use any additional heuristics you want in order to improve recognition accuracy, but this is not the main focus)

Resources for Part 1

- Demo code for several examples is available in the Code Examples folder on the course website
- JavaFX APIs for Canvas and GraphicsContext: openjfx.io/javadoc/15/
- Example code for rotateX and rotateY methods that can be called from your "rotate" method (note that argument thetaR is in radians, not degrees):

```
private double rotateX(double x, double y, double thetaR) {
    return Math.cos(thetaR) * x - Math.sin(thetaR) * y;
}

private double rotateY(double x, double y, double thetaR) {
    return Math.sin(thetaR) * x + Math.cos(thetaR) * y;
}
```

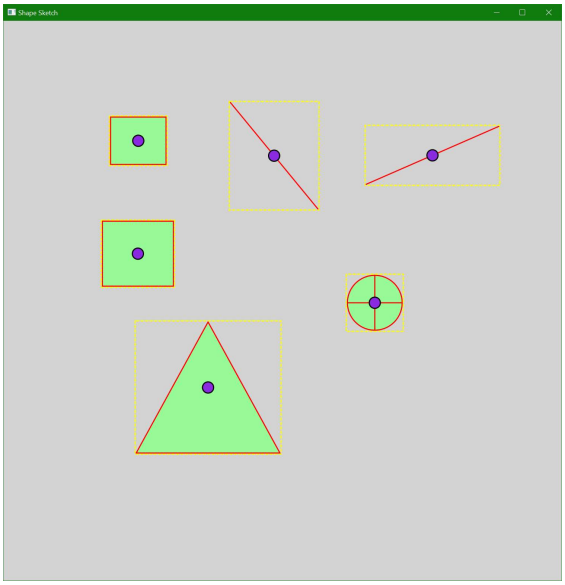
Result for Part 1: a zipped IDEA project that meets the interaction and code requirements above. To export your project, choose File → Export → Project to Zip file. NOTE: if you have fully completed Part 2, do not hand in a project for Part 1.

Part 2: Multiple selection and grouping

Add multiple selection and grouping to your system, and adapt the capabilities of the system from Part 1 to work with selection sets and groups.

Multiple selection – Interaction Requirements:

- Pressing the mouse button on an object when the Shift key is also pressed adds to or removes from the selection
- Each selected shape is shown as being selected
- If the user presses and drags on any selected shape, all selected shapes will move
- All selected shapes show a rotate handle; if the user clicks and drags on any of the rotation handles, all selected shapes are rotated (using their own centres as the centre of rotation)
- Clicking on a single unselected shape clears any multiple selection (and single-selects that shape); clicking on the background clears any selection.



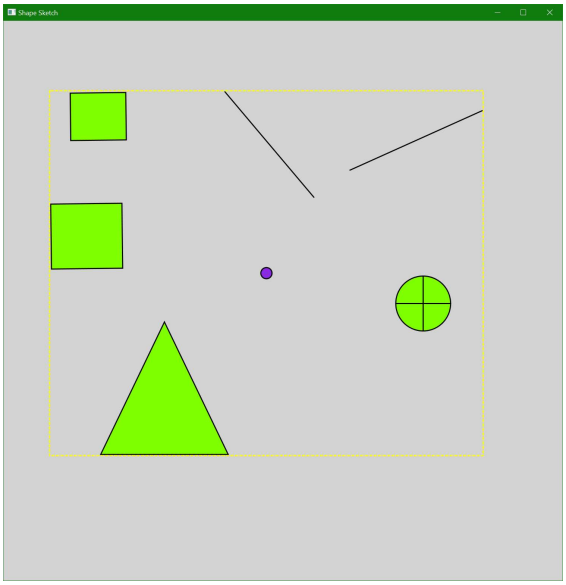
The user has selected multiple shapes

Multiple selection – Code Requirements:

- Update your InteractionModel and ShapeController classes to work with a selection set rather than a single selected shape; use a collection class such as ArrayList to store the selection set, and write a new method to add/subtract a shape from the selection
- Note that JavaFX class MouseEvent has a method "isShiftDown()" that you can use to determine whether the Shift key is pressed when a click occurs
- For any other methods in the MVC architecture that use the selection, update the methods to use a selection set

Grouping – Interaction Requirements:

- When multiple shapes are selected, the user can group them by pressing the G key
- Grouping causes the selected shapes to be added to a group, with the group now selected instead of the individual shapes; a selected group shows the bounding box of the group as a dashed yellow line, and a rotation handle at the centre of the group
- If the user presses the mouse on any shape in a group and drags, the entire group moves
- If the user clicks and drags on the group's selection handle, all shapes in the group are rotated, using the group's centre as the centre of rotation
- Groups can also be grouped (and groups of groups, etc.)
- The user can ungroup a selected group by pressing the U key
- Ungrouping replaces the group with the individual shapes that were in the group, and all are selected after ungrouping
- If there is more than one item selected, or if the selected item is not a group, ungrouping does nothing



The user has created a group with three shapes

Grouping – Code Requirements:

- Add interface Groupable (described in lectures) to allow both XShapes and XShapeGroups to coexist in the model and selection
- Change your shape classes to implement Groupable
- Add class XShapeGroup to your project; objects of this class will hold groups of shapes. XShapeGroup should also implement Groupable
- Convert the model's collection of items and the InteractionModel's selection collection to use type Groupable
- For transforms or drawing of a group, XShapeGroup should use the "delegate to children" approach described in lectures.
- Note that the only location data that a group needs to keep track of is its bounding box (and even this should be calculated from the bounds of its children).

Resources for Part 2:

- MouseEvent.isShiftDown: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/input/MouseEvent.html#isShiftDown--

Result for Part 2: a zipped IDEA project that meets the interaction and code requirements above. To export your project, choose File → Export → Project to Zip file. NOTE: if you have fully completed Part 3, do not hand in a project for Part 2.

Part 3: Clipboards and Cut/Copy/Paste

Interaction Requirements:

- If there is a selection, pressing Ctrl-X puts the selected items on the app's clipboard and removes them from the model
  - The selected items can any combination of shapes and/or groups
- Pressing Ctrl-C puts a copy of the selected items on the app's clipboard (and does not remove them from the model)
  - Only one collection of items can be in the clipboard at a time; if the user copies a second selection, the first one is discarded from the clipboard
- Pressing Ctrl-V puts a copy of the item(s) that are in the clipboard into the model
  - The user can paste the items in the clipboard multiple times
  - The pasted items become selected when they are pasted; any existing selection is unselected

Code Requirements:

- Create class XClipboard to store items that have been cut or copied. Note: do not use the system clipboard; you must create your own clipboard class
- Add handlers to SketchController to handle keypress events for Ctrl-X, Ctrl-C, and Ctrl-V.
  - Note that class KeyEvent has method "isControlDown()" that you can use to determine whether the Control key is pressed when a keypress event occurs
- Add methods for cutting, copying, and pasting to the model
- When copying to the clipboard and when pasting, you will need to make a *deep copy* of the objects, because you may be pasting them multiple times, and each paste needs to add different objects to the model

Resources for part 3:

- KeyEvent API: https://openjfx.io/javadoc/15/javafx.graphics/javafx/scene/input/KeyEvent.html
- Details on cut/copy/paste operations: Olsen text, chapter 15
- Tutorial on Java copying: https://dzone.com/articles/java-copy-shallow-vs-deep-in-which-you-will-swim

Result for Part 3: a zipped IDEA project that meets the interaction and code requirements above. To export your project, choose File → Export → Project to Zip file.

What to hand in

Note that this assignment is to be done individually; each student will hand in an assignment.

- A zip file of your IDEA project folder and a readme.txt file that indicates exactly what the marker needs to do to run your code. (Systems for 381 should never require the marker to install external libraries, other than JavaFX).
- You only need to hand in one project if that project represents everything you have completed on the assignment. However, if you have completed different elements from different parts of the assignment, or if there are bugs in one part of your solution, you can hand in multiple zipfiles to demonstrate the parts that you have completed. (If this is the case, state in your readme.txt file which features are available in which projects)

Where to hand in

Hand in your files to the link on the course website.

Evaluation

Marks will be given for: producing a system that runs without errors; demonstrating skill with MVC-based systems and 2D interactive graphics; demonstrating that you can correctly implement multiple selection, grouping, and clipboard operations.

Weighting of the parts in the overall grade: Part 1=35%, Part 2=35%, Part 3=30%.

Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.