CMPT 381 Assignment 5: Cut/Copy/Paste, Undo/Redo

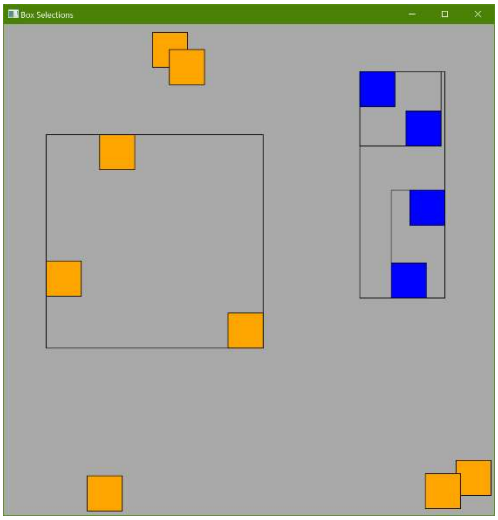
Due: Friday, March 24, 11:55pm

Overview

In this assignment, you will work with fundamental interactions in visual workspaces – cut, copy, paste, undo, and redo. You will use JavaFX to extend the Box demo developed in class, and will add several interactive capabilities to the system. You will also build on your MVC knowledge, including models that contain grouped hierarchical objects.

Part 1: Cut/Copy/Paste in a visual workspace

Using JavaFX, extend the BoxFX application introduced in class to add basic interactions such as creation of new boxes and the ability to drag selected boxes and groups. Your extended application should meet the following requirements (note that several of these are already implemented in the given BoxFX application):



***Interaction requirements*:**

- When the user right-clicks on the background, a new box is created
- When the user left-clicks on a box, the box (or the group for that box, if applicable) is selected (and stays selected after releasing the mouse button)
- When the user left-clicks on an already-selected box and drags the mouse, all selected boxes/groups move with the mouse
- When the user left-clicks and releases on the background, any selection is cleared
- When the user left-clicks and drags on the background, a rubber-band selection is started
- When the user holds down the Control key and left-clicks and releases on a box/group, that box/group is added to the selection (if not already selected) or removed from the selection (if already selected)
- Adding to or removing from the selection works both with single clicking and with rubber-band selection.
- When the user presses the Delete key, any selected boxes/groups are removed from the model

Cut/Copy/Paste requirements:

- When the user presses Control-c on the keyboard, any selected boxes/groups are copied to a custom clipboard (see below for details)
- When the user presses Control-x on the keyboard, any selected boxes/groups are removed from the current model and are placed on the custom clipboard
- When the user presses Control-v on the keyboard, any boxes/groups on the custom clipboard are pasted into the model (and shown in the view)
- Pressing Control-v again pastes additional copies of the objects
- When cut/copied and pasted, any existing group structure in the selection is preserved.

***Software requirements*:**

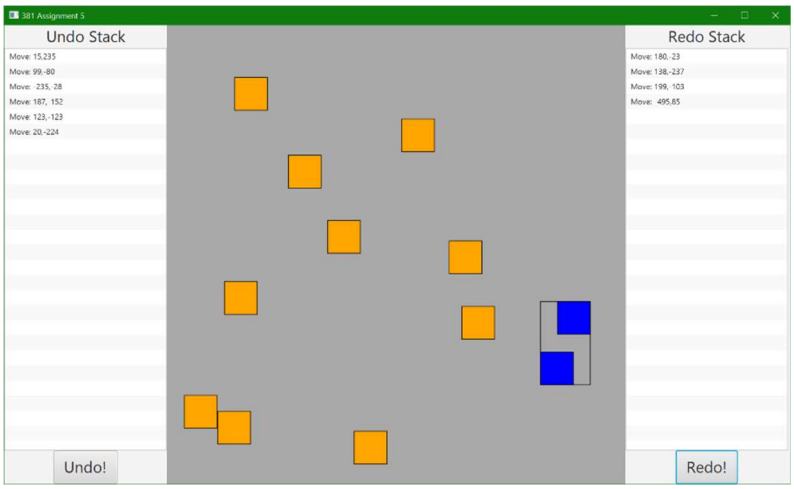
- Add code to the controller class of the BoxFX application to accomplish the new interaction requirements described above
- Add an InteractionModel class to the MVC architecture (as in your last assignment) to handle the clipboard
- Implement a custom clipboard class BoxClipboard to hold the cut/copied items. Note that you will have to return a deep copy of the items on the clipboard when you paste (i.e., not just copy the reference).
- Do *not* use the Java system Clipboard classes. Implement your own custom classes.

Resources for part 1:

- Starting application: BoxFX-groups.zip on the course moodle (Examples directory)
- 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

Part 2: Undo/Redo

In the second part of the assignment, you will extend your system from part 1 to add the ability to Undo and Redo actions in the visual workspace. You will use the Backup Undo approach described in lectures and the text, using the Command pattern.



***Additional interface requirements*:**

- Add panels to the left and right of the BoxView panel, as shown in the picture.
- Each panel contains a Label, a ListView, and a Button
- The left panel shows the current state of the undo stack, and the right panel the state of the redo stack
- Pressing the "Undo!" button executes one undo; pressing the "Redo!" button executes one redo

Additional interaction requirements:

- All box creation, box/group deletion, and box/group move actions result in the creation of a new BoxCommand object that encapsulates how to undo and redo that action
- Selection and grouping actions are *not* undoable/redoable, and it is not required that the selection state match the state of the workspace when a particular action was first carried out.
- Cut/Copy/Paste actions are *not* undoable/redoable (e.g., a paste does not count as a create action, and a cut does not count as a delete action).
- Whenever a BoxCommand object is created it is pushed onto the undo stack (and the ListView is updated)
- Whenever the "Undo!" button is pressed, the undo stack is popped and the top BoxCommand's undo() method is executed. In addition, the BoxCommand is then pushed onto the redo stack.
- When the "Redo!" button is pressed, the redo stack is popped and the top BoxCommand's doIt() method is executed. In addition, the BoxCommand is then pushed onto the undo stack.

***Additional software requirements*:**

- Implement the BoxCommand interface (following the example in the text)
- Implement classes MoveCommand, CreateCommand, and DeleteCommand that implement this interface
- Each of these command classes should include a toString() method so that you can put a string representation of the command into the ListView's list model
- In your InteractionModel, add stack data structures as needed for the undo and redo stacks (you may use the old Stack class, or the more modern Deque class)
- You may treat the Application class as the home for the ListViews (this means you should add methods to the Application class to update the ListViews when the stacks change)

This assignment is to be completed individually; each student will hand in an assignment.

What and Where to hand in

- JavaFX: a zip file of your NetBeans project folder for **either** part 1 (if that is as much as you have completed) **or** part 2. If you have completed part 2, you do not have to hand in a project file for part 1.
- 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).
- Hand in your two files (one zip and one readme.txt) to the link on the course Moodle.

Evaluation

Marks will be given for producing a system that meets the requirements above, and compiles and runs without errors. Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.