Homework 5: Undo using Command Objects

Assigned: 10/25/2022; Due 11/3/2022 at 3:05pm ET

Overview

In this assignment, you will be adding **Undo** to the graphical editor using the **linear undo model**, implemented using the **Command Object pattern**. Since there was a high variance in students' implementation of homework 3, and we didn't want you to have to convert your code into React, we have created a reference implementation of the editor in React, **which you are required to use**. Note that this version **removes the canvas** and the delete-all button (and does not include the optional line drawing), since we felt these would not contribute to your learning of undo.

As with previous homeworks, you are allowed to use certain libraries for homework 5, such as those used by our reference implementation of homework 3. In particular, allowed libraries include React, the ones that come with Create-React-App, ReactRouter, ReactStrap, and **React Context API** (which is a relatively newer API which is becoming a popular/standard way to pass data across components. Note: we recommend you read the https://reactjs.org/docs/context.html documentation to help with understanding the supplied code.). The icons we used on the buttons came from the "react-icons" package. If you implement the extra-credit feature of having a list of command objects on the right, you might find the react-scroll-to-bottom package useful.

However, you may ***not*** use libraries that implement the command object pattern or Undo -- you must implement that yourself. If you have another package you would like to use, please ask on Piazza.

The user interface for homework is essentially identical to homework 3, with the added buttons for **undo** and **redo**, and removal of the Canvas and delete-all (and line) features. We include pictures of what the base looks like, what our reference implementation looks like, with the command list on the right, and we also provide the hw5 video to see the required behaviors for undo.

Resources

Michael Liu (PhD student) and Clara Cook (last year's TA) created an implementation of homework 3 in React, which you are required to download and use. It is in the material you will download. **Note that you need to *modify* the base implementation, not just add to it. In particular, you will need to *edit the code* in all the places where objects are created, modified, or deleted to instead use command objects for those operations**, so they can be undone.

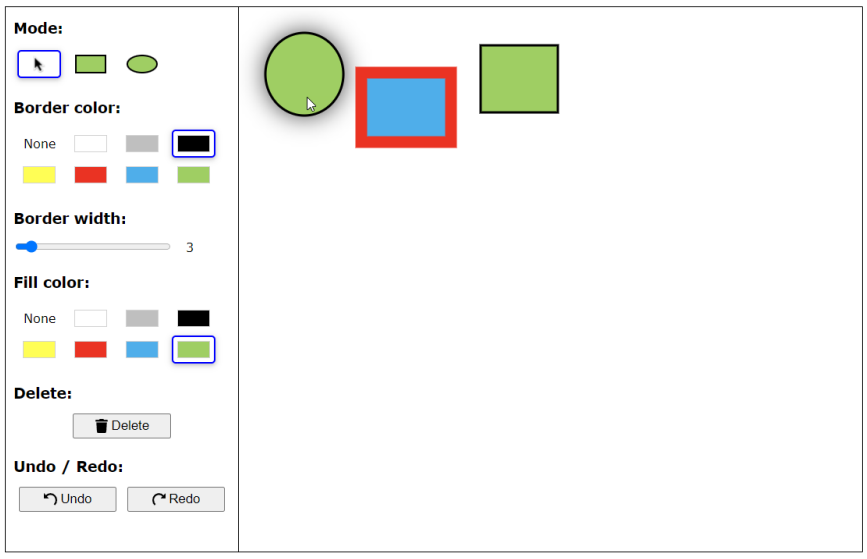
We have created a set of JavaScript specification files for how command objects *must* be implemented, and then a non-working outline of how a command object for changing the fill color might start off. You should use these to structure your implementation of command objects and linear undo. These were discussed in lecture 14 (which continued into the lecture 15 video), and are also included in the download.

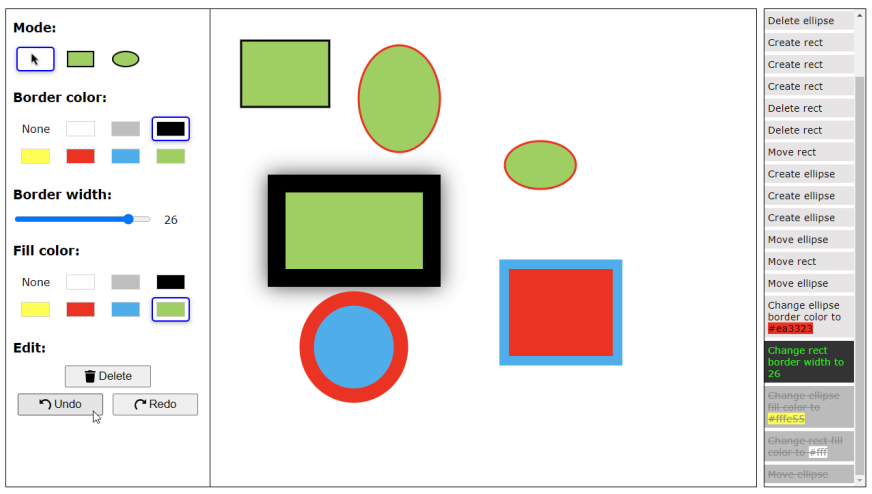
You will download a new project for Homework 5 from Github, like for the previous homeworks -- the exact location is on Canvas in the instructions for Assignment 5. The download will contain the following starter files:

- **05631-hw5-base** - this folder contains all the code that implements the graphical editor using React. You *must start from this code*, and add undo handling to it. It also contains the following:
  - **CommandObject.js** and **ChangeFillColorCommandObject.js** - these files are in the 05631-hw5-base folder in the subdirectory: 05631-hw5-base/src/shared/commandObjects/ and provide the structure and an example for how you should implement your command objects.

In addition, we have supplied the following for your reference:

- HW5-example.mp4 - a movie showing how the homework should look and behave. You can also see it by clicking on it here.
- HW5-base.png - a picture of what the interface starts off like from the base version.
- HW5-command-list.png - a picture of what HW5 might look like after a few commands are executed, and then three commands are undone.





Detailed Requirements

We have supplied a movie to define the required behaviors. As with previous homeworks, you should be careful to meet the specification below for full credit. If something is unclear about how it should work, or you want to deviate from the defined behavior, you might want to first ask on Piazza.

- All editing operations need to be **undoable**, in the conventional manner. By "conventional", we mean for the undo to be similar to the way that PowerPoint works. The details of the required behaviors are specified below and shown in the hw5 video. Undo should basically restore the drawing as if the undone operation had not happened, and then select the affected object.
- The user must be able to do **multiple undo** operations in a row, all the way back to the beginning (the linear undo model).
- After any undo, the user must be able to use **redo** to undo-the-undo, and restore the drawing to the way it was before the undo happened. If there have been multiple undo operations, then redo should work that number of times as well.
- If after the user performs one or more undo operations, and then does an operation that is neither undo nor redo, then the rest of the previously undone operations should be discarded and not available (the standard way that the linear undo model works), and the new operation should go on the undo stack.
- You need to define classes that extend `CommandObject` for each type of operation (create, delete, move-object, change-fill-color, change-line-color, change-border-width). Each of these should store all of the needed information to undo or redo the operation in the command object itself.
- The buttons for Undo and Redo should be greyed out when those operations are not available. (Undo should only be greyed out when there are no operations at all, and redo should be greyed out whenever there are no undone operations that could be redone.)
- The operations that need to be undoable include:
  - **create** - undo makes that object disappear. Nothing will be selected after an undo of create, since nothing would have been selected before the create happened. (Note: PowerPoint re-selects the one that was selected directly before the create, independent of what was selected before the undo. So this is one place that our design will *differ* from PowerPoint.) However, redo of a create operation should cause that object to be selected (and the application to change to selection mode), even though creating objects does not select the object after it was created.
  - **change fill color**, **change border color** - undo makes the color go back to what it was before the change. If the user clicks on a bunch of colors while an object is selected (so it changes colors multiple times), these should all be individually undoable. The object affected by the undo should become selected, and the palette should reflect its current values.
  - **change border width** - even though the object changes continually while the Border Width slider is dragged around, the undo command should undo the whole operation back to before the user started dragging the slider. For example, if the border width of the selected object is 3, then the user clicks on the slider and drags the slider (while holding the mouse button down) to 4, then 5, then 6 up to 10, then down to 9, 8, and so on down to 2 and then lifts the mouse button, the selected object will have had its border take on all of those widths, but the end result is that the border changed from 3 to 2, so only one command object should be created, and it should restore the border back to 3, and redo of that would make the border be 2. (So there is a single undo operation for the change border width.) If the user changes the border width twice in a row, by mousing down on the slider two times, those should be separate commands and separately undoable. As with change-color, the object that the operation affected should become selected.
  - **move** - undo of moving an object makes the object go back to where it was before the move. As with the border width slider, only the original position (before the move) and the final position (after the mouse button is released) are relevant for undo and redo. The affected object should become selected.
  - **delete** - undo of delete makes the object come back, and become selected, and redo makes it go away again and nothing be selected. Note that user can change the selection manually after the undo, and then perform a redo, in which case that redo should still undo the delete of the *original* object (whatever was deleted the first time is deleted again independent of what is selected when the redo occurs).
- Note that normal changing the selection by clicking around on objects is *not* undoable. That is, no command objects should be allocated for just changing selection or clicking on the background to change there to be no selection. If the selection changes due to undo or redo, then the palette items should change to reflect the values of the object that becomes selected as a result of the undo or redo, just like if the selection was changed by the user.

- Similarly, changing the drawing mode (arrow, rectangle or circle), or else changing the current colors or line width when there is *no* selection, should not affect the undo stack, since nothing changes in the drawing. Therefore, these changes in the palette are *not* undoable. But as mentionend above, the palette might be affected by undo/redo if the selection changes.
- Aborted operations (operations that are started but aborted using ESC so they don't do anything) should not affect the undo stack (since they didn't actually change anything). Thus aborted operations should not affect undo operations at all.
- Undo and redo should always change the global mode to be selection (in case it was create rectangle or create ellipse mode beforehand). For example, if the user is in selection mode, selects an object, changes its color, then changes mode to rectangle mode, then does undo, the mode should be changed back to selection mode so the object whose color was undone can be selected. It is fine to just always change to selection mode with undo and redo, even though nothing needs to be selected for undoing a create or redoing a delete.

Extra Credit

- We accidently left in the line drawing and line editing operations in the reference implementation of homework 3 in React, which are not required for this homework, as shown in the pictures and video above. You can just comment out the line code, or you can add support for undoing all the line drawing, deleting and editing for up to +5 points extra credit.
- Make typing ^Z (or command-Z) and ^Y (or command-shift-Z) work as accelerators for Undo and Redo respectively. [Up to +2 extra credit - +1 each]
- Add the command list panel to the right of the drawing area, as in the HW5-command-list picture and the HW5-example video. For full credit, it should show the current operation, discarded operations should be removed (due to doing a new operation after some undos), and undone operations should be greyed out. [Up to +5 extra credit]
- Add the **repeat** operation for all editing operations. This should work as specified in lecture 14. In particular:
  - The label and icon on the "redo" button should change to be "repeat" when appropriate.
  - Repeating the previous *change-fill-color, change-border-color,* or *change-line-width* operation should do the same change to the currently selected object, if something is selected, otherwise it is not enabled.
  - You should be able to repeat the *create* command, which will create a copy of the previous object (same size, colors and border width), but offset by 10 pixels in its left and top positions.
  - Repeat of delete should delete the currently selected object.
  - It is *not* necessary to make repeat work for move operations. (PowerPoint doesn't.)
  - The repeat operation should appear on the undo stack so it can be undone in the normal way. If you support the command list panel, it should have a reasonable name, like "repeat of change line border color to #ea3323".
  - The repeat button should be greyed out when it cannot be used (if the previous operation was a move, for example).
  - [Up to +15 extra credit, out of 100]
- Add 4 "**nudge move**" commands, one in each direction, that moves the selected object by a few pixels with each press of the keyboard arrow key in that direction, and an undo of the nudge move. For full credit, you should *coalesce* multiple nudge operations into a single undo, in the same way as PowerPoint operates. [Up to +8 extra credit, out of 100 (+2 points for each direction)]
- Add support for **selective undo** and **selective repeat** using the direct inverse model. This requires that you implement the command list panel extra credit item *and* the repeat extra credit item above, and then also support:
  - Selecting a command in the command list panel by clicking on it.
  - Two new buttons for "selective undo" and "selective repeat" in the Edit part of the palette, which are enabled when those operations are possible.
  - Clicking on the buttons cause the selected command in the command list to be selectively undone or repeated, and then puts that operation onto the undo history (so it appears at the end of the command list). The name of the operation should be something reasonable.
  - [Up to +15 extra credit, out of 100] -- *This is a challenging one to get right, but you can get up to +35 points if you do all the parts (command list + repeat + selective)*

Grading

Since this homework has lots of parts, some students may want to know what each part is worth. Here is a summary breakdown: total = 100 points:

- 5 points - readme
- 95 points - implementation
  - 20 points - overall implementation and good coding style
  - 75 points - undo / redo behaving correctly
    - 10 - correctly adds and removes items from the undo stack, and grays out the undo and redo buttons accordingly
    - 15 - create undo / redo
    - 50 - 10 each for 5 other undoable operations

Turn In

Please have a README file in your subfolder (as in previous homeworks), which can be in plain text, Microsoft Word, or pdf format, which should contain:

- Your name and Andrew ID (please include both!) at the top of the file
- The URL of your Github Repo.
- Any extra credit work you did, and what the user interface is to make it happen..
- A discussion of what you found most difficult or confusing to implement in this assignment.
- Anything else of interest in your software design or implementation.

Please upload this README file to Canvas by 3:05pm ET on 11/3/2022.