Selection Box and Dragging Multiple Items Postmortem

FirstI had to thoroughly search joint documentation trying to find a way to create a simple box, but couldn’t find anything.

It took a while of searching, trying different demos/code samples to figure this out however. So then I knew that I would have to draw directly to the javascript canvas. I tried to draw to the normal raster canvas, but it didn’t work. After much trial and error I finally figured out that Joint.js was drawing to a svg canvas instead. Had to insert an html element directly into the document object model.

After doing that boxes are being draw onto the screen, but are only being drawn properly when dragged from top left to bottom right. After some painstaking debugging I determined that svg elements must have a positive width/height and always have to be drawn from top left to bottom right. So what if the user drags from bottom right to top left? Nothing. The solution to this problem was relatively simple, but the time consuming part was isolating the problem. The solution was to create a function that converts two arbitrary points into SVG compatible rectangle details (width, height, starting point).

At this point a box is drawn onto the canvas as a mouse is dragged, but when the canvas is zoomed in/out the box isn’t drawn at the appropriate coordinates and was too large/small. After debugging for a while I determined that the coordinates that the box was being drawn with were relative to screen, since they are mouse coords. However, the when Joint.js was zooming a graph in or out, it was actually zooming at the SVG canvas level. To solve this I track the relative zoom of the canvas and scale the coordinates that the SVG selection box is drawn with appropriately. There was another issue, when the canvas was panned the selection box was drawn at the wrong offset coordinates. This was solved really similarly to the previous issue. Now we have a robust selection box that is drawn with a cursor while being dragged.

While graph items were currently selectable, existing object strokes / styling information were being overwritten and lost in the selection process. So I redid how whether an object is selected is stored and displayed. A major help for this process was when I found a highlight() method in the Joint.js documentation, accessing this method was a little tricky though. After some headbanging, I figured out that it was because the graph only holds some attributes for objects (width, height, text, etc…), but highlighting and styling information is held in the “Paper”. The fix for this was to query the paper for the view that corresponded to the graph data item that we alread had, and to set the view as highlighted.

Next we want to detect a drag, and then apply a translation to all highlighted elements by the proper (scaled and relative) amount. To detect the drag we have to register an event handler with the paper. After detecting the drag motion I attempted just to call the translate() method on the selected items. However, now the item that we are dragging to move the group around is being moved twice as much as the other items in the group. After much troubleshooting I figured out that the Joint.js paper’s internal event handlers are probably attempting to move the item while we are doing the group translation. Essentially the object was being dragged simultaneously by two event handlers.

After scouring the Joint.js documentation for a way to disable the default event handler nothing turned up. It was actually difficult to tell which graph element is actually being dragged from within our Joint.js event handler as well. So I attempted to undo our second drag with yet another drag handler, which would perform a negative translation. This would, in effect undo our other drag handler. Now our dragged item is moving properly, but the other items in the group are doing weird stuff. Jumping around and “jittering”. I looked into the possibility that I could interrupt the initial event handler and consume the event, which would prevent the double dragging. However the paper’s event handlers are always registered before, and thus have higher priority than ours. Around this point the two Michaels from the implementation team spend an hour or two with me trying to figure out a solution. But nobody else could really figure out anything else to try and we ran out of time to figure out other solutions.

That’s why drag selection and multiple item movement isn’t in the final product, even though it was 95% done.