Homework 2: Handling Input Events

Assigned: 9/9/2021; Due 9/21/2021 at 3:05pm ET

Overview

In this assignment, you will be creating a placeholder (temporary) web page with advanced input handling. The events will be just manipulating html `div` elements, which will be replaced with handling events over graphics objects in homework 3. The goal of this assignment is to learn how to handle multiple kinds of events that may interact with each other, and deal with the required interactions among them. Like homework 1, you must write this assignment without using external packages - just regular JavaScript (and html and css).

You should build this homework as a new page in a subfolder, linked off of the CREATE YOUR OWN link on the "Scotty Shirts U Illustrate (SSUI)" web site from homework 1. We have supplied a starter html page and its CSS page. Please put the code for homework 2 into a *new* JavaScript file called `input.js`, all of which will be in your new subfolder.

Hint: Naturally, your touch event handlers will not work on a regular laptop, so that will most-likely need to be tested using a smartphone or tablet (unless you happen to have a touch screen on your laptop). The netlify link you created for your repository might be helpful for running your page on that device.

Resources

This homework will only have 1 page, which we supplied as index.html - you should put this in a subdirectory and link to it from your CREATE YOUR OWN button from homework 1. We are also supplying the inputstyle.css file that goes with it. We supplied an empty input.js file that you will put your code in. You will download the SSUI-hw2.zip file which will contain the following starter files:

- SSUI-hw2.zip - contains all the starter files and images. Download this file to get started.
- index.html - the main page for homework 1. It displays 3 red rectangles (divs) which you will write code to manipulate.
- inputstyle.css - style file for this homework.
- input.js - empty file you will fill in.

In addition, we have supplied the following for your reference, also in the zip file:

- HW2-example.mp4 - a movie showing how the homework should look and behave. You can also see it by clicking on it here.
- input-test - a folder containing a little program that shows what events are generated, it contains its own index.html and index.js. You can also just run it from here: https://www.cs.cmu.edu/~bam/uicourse/05631fall2021/HW2/input-test/index.html.

Debugging on Phones

A key challenge when programming touch interactions is how to test them. The Chrome debugger on PC or Mac has a mode to pretend the mouse is a finger (the icon with the small and bigger rectangles, next to the inspect icon), but this does not support testing of 2 (or more) fingers. If your PC or Mac has a touch screen, then the usual mechanisms will work fine, but if not, and you need to use your phone or tablet, then you need to use special mechanisms for debugging on your phone. A professional phone developer has tools which make this easy, but we found something simpler: a way to see the Console output on the Chrome browser for your phone. This works on either Chrome for iPhone or Chrome for Android. Thanks to Clara Cook (TA) for figuring this out:

For Android Phones:

1. Go to Settings > About Phone (which is all the way down at the bottom) > Build Number (On Later Devices, may be under "Software Information")
2. Hit this Build Number Button repeatedly until you are in developer mode (about 7 times)
3. On computer, in chrome search bar, go to "chrome://inspect/#devices"
4. Make sure "Discover USB devices" is enabled
5. Open Netlify App on your phone in chrome
6. Hook up to computer using USB
7. Return to computer Chrome screen and refresh
8. Choose the "inspect" link from the correct website from the list of remote devices that should appear in your computer browser
9. This should mimic your phone screen! Make sure you can find the console!

For IOS:

1. On your phone, make sure you have chrome downloaded.
2. Open chrome and go to "chrome://inspect"
3. Hit the button that says start logging.
4. Go to your netlify app on a new tab inside your phone's Chrome
5. Toggle between the two screens to make sure your app is being logged!

If someone finds a better way, please let us know!

Detailed Requirements

The supplied html file contains 3 "div"s which are what your code should operate on, however your code should work no matter how many divs there are (it shouldn't be tied to these specific 3 divs), but you can assume they will all be of the same `target` class. Various mouse or touch events should cause different things to happen to the divs - they become selected or de-selected on clicks, they move if the user presses down and drags (or if the user double-clicks on them), and two fingers can be used to change their size. However, it gets more complicated to keep these behaviors from interfering with each other, and to support error cases and aborting. In particular:

- **mouse-click on a div** - causes the div to be selected, and it changes color to be blue. If another div was selected, it should be deselected.
- **mouse-click on the background** (on workarea) *not* on any objects - deselects the selected div, if any.
- **mouse down on a div and move** - causes the div to start moving, following the mouse, until mouse-up. It should not change color (this way of moving should not change the selected div).
- **mouse double-click** - causes the div to be selected (change color), and *also* starts a mode where this div follows the mouse (even though the mouse button has been released). The div should stop following the mouse on the next mouse up event. While in this mode, all other mouse behaviors on the divs should be suspended (so, for example, clicking on a different div will *not* select it). This is a common accessibility feature for people who have difficulty holding down a mouse button.
- If the user hits the keyboard **"ESC" key** *while* the operation is *in progress,* then the moving or growing should *abort*, and the div should go back to the way it was before the operation starts. Hitting "ESC" when an operation is *not* in progress should do nothing. (Obviously, this mainly applies to mouse behaviors, but if you have a keyboard on your tablet, or a touch screen on your laptop, then this should work with moving and changing size from touch events too.) Aborting a move or grow should not affect which object is selected.
- **one-finger tap and double-tap** - as explained in class, these generate mouse click and double click events, so they should do the same thing as mouse-click and mouse-double-click.
    - After double tap on the touch screen, the div gets to be in "following the finger" mode. When you do touchstart, the div follows the finger *even if the touch is not on the object*, until touchend. Touchend does *not* stop the mode, however, so the next touchstart *continues* to move the same div (again, even if the touchstart is not on the div). This mode only ends with a click action (touchstart and touchend quickly in the same place).
    - If the user puts down another finger *while* the first finger is already moving the object in this "following the finger" mode, then this should **abort** - return the object to its orginal position before the move started and stop being in this mode, just like hitting the ESC key on the keyboard. It should not affect which object is selected.
- **one-finger touch drag** - This should have the same behavior as mouse-drag (move the object around with the finger).
    - If the user puts down another finger *while* the first finger is already moving the object, then this should abort - return the object to its orginal position before the move started, just like hitting the ESC key on the keyboard. It should not affect which object is selected.
- **two-finger touch** - If the user puts two fingers down at approximately the same time, this should change the size of the selected div in place, without moving that div. The size should approximately follow the two fingers, while keeping the center of the div pinned in place. The requirement is that this work horizontally, with vertical size changing being extra credit (the video shows both). That is, moving the fingers will change the left and right sides of the div to be as far apart at the fingers are. If nothing is selected, then using two fingers should do nothing. The change-size should stop when the two fingers are released.
    - If one finger is released but the other stays on the screen, this should stop the size change as it last was (just stop growing but *not* abort), since this inevitably happens when one lifts two fingers - one finger lifts off first then the other.
    - If the finger goes back into contact before the other finger lets up, the div can go back to changing size.
    - If a third finger is put down on the surface, this should abort the change size and go back to the original size before the change size started. All further finger activity should be ignored until all the fingers are removed from the surface.
    - Your code should enforce a minimum width and height for the div. Any attempts to make it smaller should just be ignored, but the change size continues, in case the user moves their fingers further apart again.
- Your code should work for any number of div's - not just the 3 in the provided html file. You can assume all the div's will have the same structure as those provided (they will have have the same `target` class).

Extra Credit

- As shown in the video, support changing the size with two finger touch in *either* horizontal or vertical directions, depending on the orientation of the fingers when they both touch the surface, as shown on the video. For example, if the two fingers are approximately horizontal, and then the fingers will change the left and right sides of the div. Alternatively, if the user puts their fingers approximately vertical, then this should change the top and bottom. *[Up to +5 extra credit, out of 100]*
- Add in a behavior where pressing down in the background, and enclosing an area while the mouse button is held down selects a div that is entirely within the bounding area, like in conventional drawing programs like PowerPoint. (If there is more than one, you can select any one you want.) Preferably show a bounding box to provide feedback for the current bounding area. *[Up to +5 extra credit, out of 100]*

- Add an event handler for a *different* kind of event (should be of a different form than the above, so it doesn't count as extra to handle something like touchcancel). Some examples that would count are (see large list of events): window resize event, scroll wheel, devicemotion (accelerometer), etc. Do something interesting to the divs as a result of those events. *[Up to +10 extra credit, out of 100]*

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
  - 10 points - overall implementation and good coding style
  - 60 points - mouse part
  - 25 points - touch part

Turn In

Please have a README file in your subfolder (different from the README for hw1), which can be in plain text, Microsoft Word, or pdf format (*NOT* .md this time), which should contain:

- Your name and Andrew ID (please include both!) at the top of the file
- Your site's URL on Netlify, and the link to your Github Repo (these will probably be the same as for homework 1, but please put them in your README anyway).
- A description of your design, especially how you coordinated the multiple event handlers. That is, what strategy did you use in your implementation to control what behavior happened for each event.
- Any extra credit work you did, and what the user interface is to make it happen.
- Anything else of interest in your design or implementation.

Please upload this README file to Canvas by 3:05pm ET on 9/21/2021.