Isaac White
iaw2105
User Interface Design, Fall 2013
HW3

The guiding principle for my implementation of a gesture based video playback system is to try to keep things as simple as possible. To this end, I don't think a gesture based system would have been best; there are a lot of functions that a user might not be able to easily remember that would be fine to include in a menu. I tried to keep my gestures as consistent with the action they are related to as possible (recognition rather than recall), but since the requirement was to only interface with the system using gestures, it makes it more difficult for the user to interact.

Where possible, I tried to reduce the gestures to a single movement. A single line in a certain direction that carried meaning in relation to its goal, (such as volume up being an up arrow, or skipping forward being a right arrow), and in other places I adopted the basic conventions of other, non-gesture based video players, like click to pause or unpause. I am hoping that this reduces the need on the user to remember complicated gestures. Even the gestures that were more complicated, I tried to imagine an action in physical space that would produce the same result (like the gesture for resizing the video is similar to grabbing the video by an area and either pulling it outward or inward, if you think of the left edge as the inner edge). Likewise, to increase the speed of the video, you make a circle in a clockwise manner (like a clock), and to go slower you move in a counter clockwise motion.

I added an additional gesture to set the volume back to the maximum value, since the user interaction with the system seemed to necessitate fixed size increases or decreases. For volume, a good amount seemed to be 10%, while for speed 50%

seemed to be good. For skipping forward and backward, 10 seconds seemed to be a good amount of time. The reason fixed size increases were needed was because as discussed in class, using a fixed positioning system without visual feedback of anticipation (like trying to tap on a tablet without a screen underneath and hitting a target matched to on another area), is difficult if not impossible to accomplish successfully. Since the gesture recognizer didn't handle points in real time, the solution for this was to let the user interact with the system in fixed size changes, in sizes that seemed to be the most useful based on the context. I would have preferred to use a slider for this, since I'm really a fan of the way sliders work, particularly on iOS devices allowing for different speeds of seeking depending on the distance away from the slider, but since no interaction was allowed with any buttons or sliders (except for help), this didn't seem to be an option.

Despite the user not being able to interact with buttons or sliders directly, I still wanted to make sure that the user actively had a way to view some simple help documentation and to get feedback about the action that had just been performed. To this end, I added a function that would display messages near the bottom of the window, indicating whether an action was performed successfully or not, the subsequent status of the system, as well as a fade out to not inhibit the viewing of the video.

For maximum availability of the gesture area, I decided to make the entire browser window allowable gesture area. The advantages of this can be seen easily if you look at the demo page for $1 recognizer. Since the canvas area is of fixed size on this page, it's very easy to start a gesture and accidentally finish outside the gesture

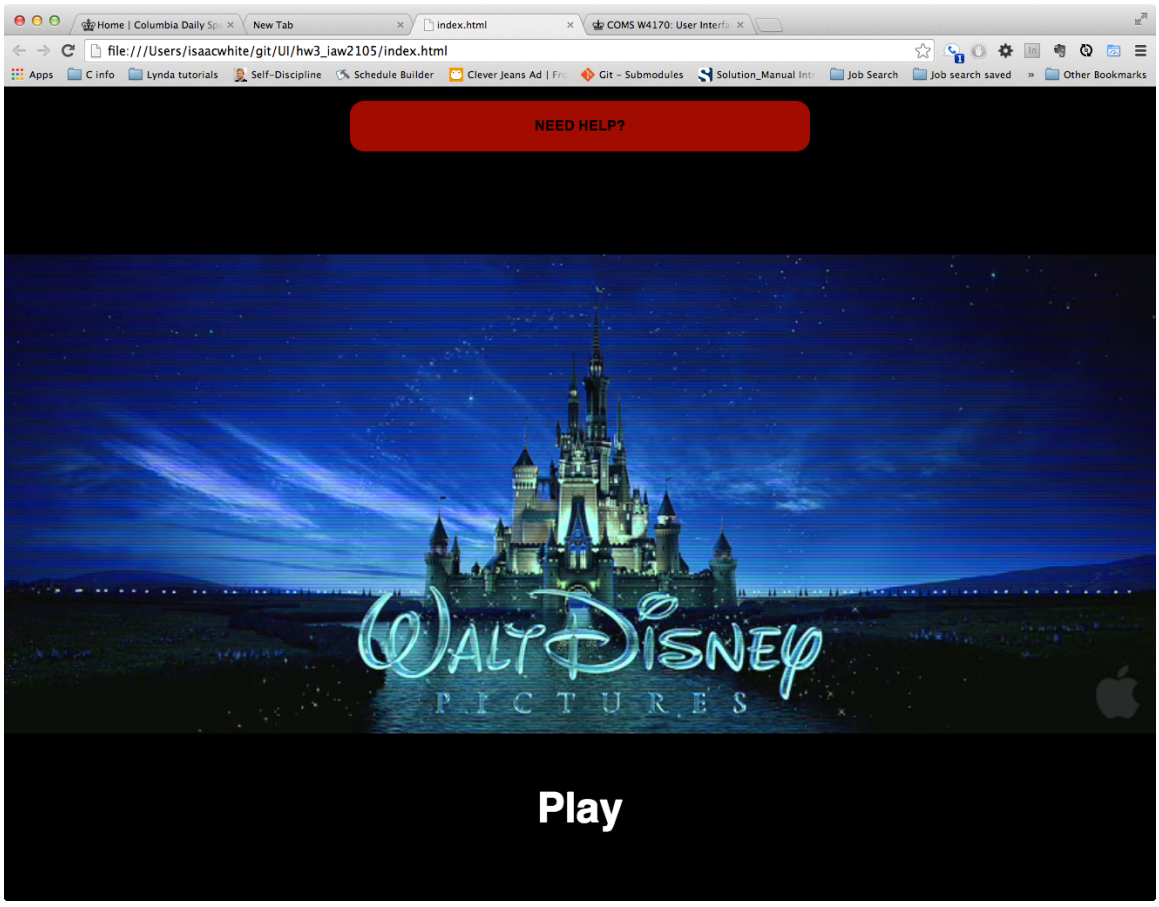
area, invalidating the gesture completely. By increasing the size of the gesture area to the entire window, it makes drawing the gestures easier by not having to make them too small, and also prevents many mistakes by accidentally going outside the gesture area.

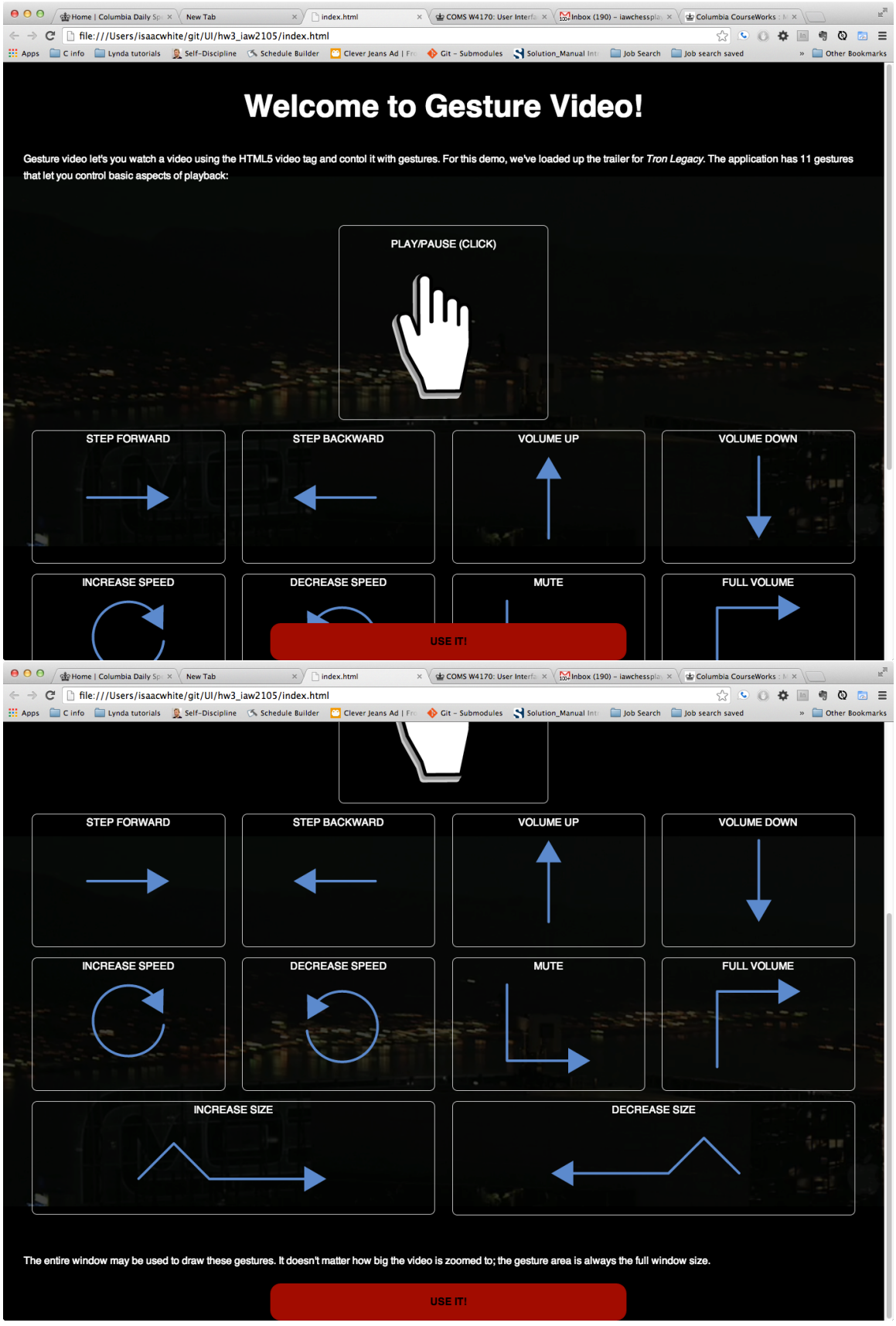
After my initial attempts with gestures and the $1 recognizer, it became apparent that my gestures would need to be more distinct. The $1 recognizer is rotation invariant, which means it cannot distinguish between gestures which appear different to the user but have rotational symmetry. This is especially a problem with recognizing simple line gestures, which the $1 recognizer also tries to scale and stretch to match other gestures. Because of this, running lines as gestures through the $1 recognizer could match against almost anything (the axis that doesn't have a lot of change gets stretched until the noise resembles something random), making recognition poor. Nevertheless, line gestures are particularly friendly to users that have a sense of forward, back, up, or down, and for that reason I chose to use them anyway while working around the $1 recognizer's limitation. By including lines and simple clicks (not recognized successfully by the $1 recognizer), I was able to reduce the number of more complicated gestures to only 5, which also control functions that are less often needed.

In resizing the video, by acknowledging that stretching the video is not something that should be supported, I was able to merely let the user adjust the width of the video while automatically constraining the height. This eliminated the need for more complicated diagonal gestures as well.

In conclusion, my application tries to be as minimalist as possible while providing an easy to learn user interface within the constraints of the assignment. A help page is frequently visible that lists all the possible commands (by diagram), as well as their functions. The video player provides immediate feedback upon completing an action, which enables users to confirm that they performed what they were intending to do, or realize they did something wrong and seek help via the help button that is visible on the page.

SCREENSHOTS:




Home | Columbia Daily Sp
New Tab
index.html
COMS W4170: User Interfa
Inbox (190) - iawchessplay
Columbia CourseWorks
file:///Users/isaacwhite/git/UI/hw3_iaw2105/index.html
Apps
C info
Lynda tutorials
Self-Discipline
Schedule Builder
Clever Jeans Ad | Fre
Git - Submodules
Solution_Manual Intr
Job Search
Job search saved
Other Bookmarks
Welcome to Gesture Video!
Gesture video let's you watch a video using the HTML5 video tag and contol it with gestures. For this demo, we've loaded up the trailer for Tron Legacy. The application has 11 gestures that let you control basic aspects of playback:
PLAY/PAUSE (CLICK)
STEP FORWARD
STEP BACKWARD
VOLUME UP
VOLUME DOWN
INCREASE SPEED
DECREASE SPEED
MUTE
FULL VOLUME
USE IT!
INCREASE SIZE
DECREASE SIZE
The entire window may be used to draw these gestures. It doesn't matter how big the video is zoomed to; the gesture area is always the full window size.
USE IT!

The application is controlled primarily from the video playback screen by performing gestures anywhere within the window. All possible gestures are documented on the help page presented to the user upon first opening the application. A button label “Use it!” is held fixed against the bottom edge of the help screen so it may be accessed immediately without needed to scroll past the instructions.

SKETCHES:

Sketches for the application and associated gestures can be found on the following pages of this PDF. The biggest thing that changed within the application throughout the development was the gestures. These changes were mainly motivated by difficulties with the $1 recognizer’s ability to distinguish gestures from one another, but also encouraged the simplification and consolidation of unnecessary gestures, which ultimately improved the user interface.

Embody the function in the gesture

play

pause

toggle

seek forward

seek back

simplify

increase play speed

decrease play speed

Mute

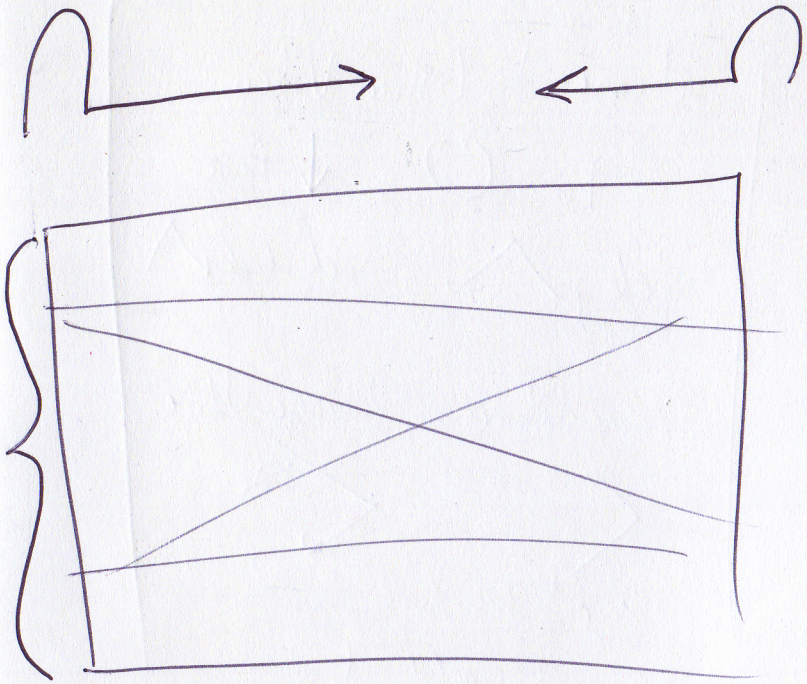
Increase Volume

simplify

decrease Volume

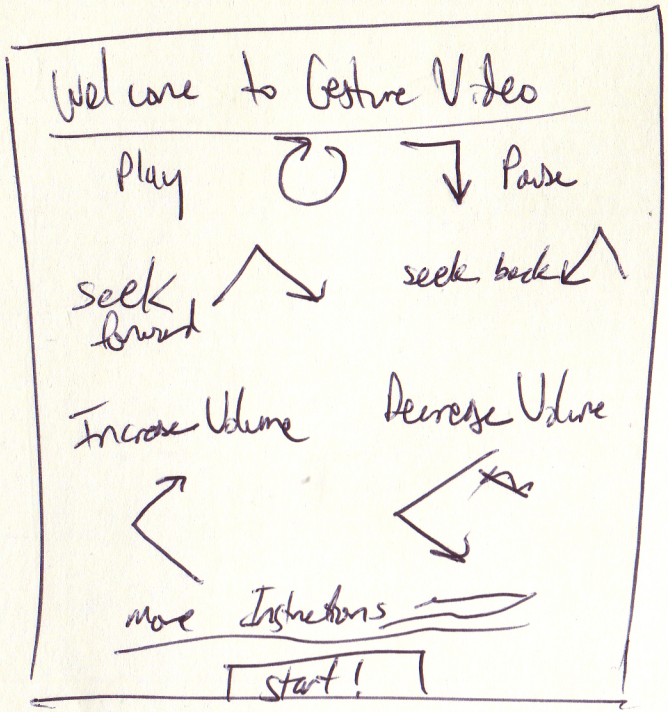
Resize larger

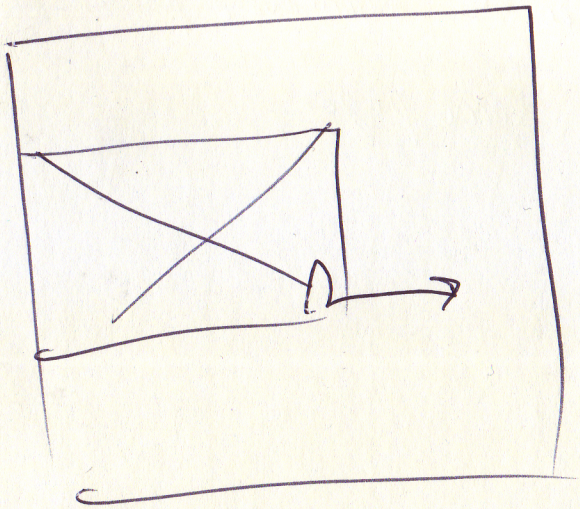
Resize Smaller

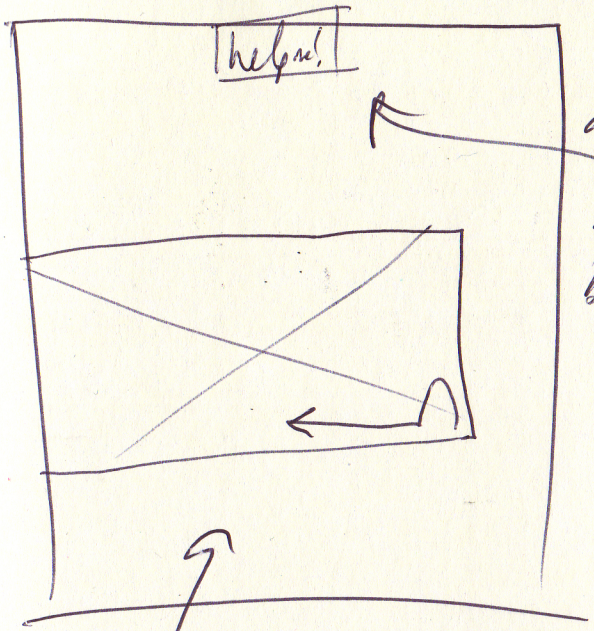


Use entire screen: avoid problems like demo page

Help page







Maybe include status message