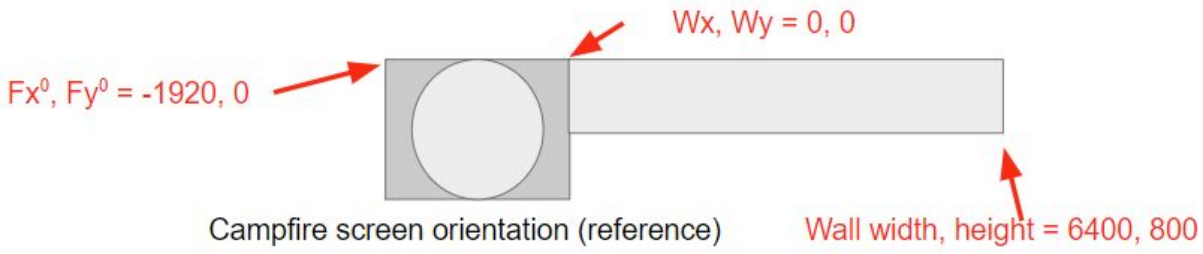
Mouse Wrangler Documentation

Advisors: Professors James Hendler and John Erickson
Created by: Tommy Fang
Spring 2018

Table of Contents

IDEA Rensselaer Institute for Data Exploration and Applications



Introduction:

The Mouse Wrangler is an Electron app that is intended for usage with Windows. The application is an JavaScript object stored on **npm**. You can see the documentation here or the actual code located on GitHub to see the functions. The displays of the Operating System must be set up as above.

The main display is the wall screen, so the origin of the wallscreen will be 0,0. The main floor screen is located left of the wall screen in Window's perspective. The current implementation works with displays of variable sizes, the most important factor is keeping the wall origin at 0,0 and the floor screen to the left of the wall. It is important to check the scaling of apps on the Display properties menu. It should be at 100%, if the scaling is changed, it will affect the coordinate system.

We can begin a new project by creating a blank electron script, which is essentially JavaScript. The line require('@campfirehci/mousewrangler)(args) is all that is required in this new file.

Installation:

1. Clone repo from https://github.com/TheRensselaerIDEA/campfire-hci
2. Open cmd and cd to path/to/campfire-hci directory.
3. Type “npm install” to install the node modules directed by package.json.
4. Type “electron .” in the directory of the project folder or “electron campfire-hci/exampleproject”
5. If there are npm rebuild/compatibility errors with electron, run this command in the directory of package.json, where target should match the local electron version.
    a. `npm rebuild --runtime=electron --target=1.8.4 --disturl=https://atom.io/download/atom-shell --abi=48`
6. **Installing Dependencies**
    a. **MSBuildTools** - used to build **win-mouse**
        i. http://landinghub.visualstudio.com/visual-cpp-build-tools
    b. **win-mouse** - Specifically creates an event listener for mousements.
        i. https://www.npmjs.com/package/win-mouse
    c. **Python 2.7**
        i. set PYTHON path variable to C:/../path/to/Python27.exe
        ii. OR npm install --global --production windows-build-tools
    d. **Electron.js** - Platform used to store wrangler script and interact with
        i. https://electronjs.org/
    e. **Node.js** - https://nodejs.org/en/
    f. **Node-gyp** - Compiles C drivers used by win-mouse to listen for mouse inputs.
        i. npm install --global node-gyp
    g. **Robot.js** - module used to move mouse positions for user.
        i. See how **Robotjs** interacts with node-gyp here: https://www.npmjs.com/package/robotjs

var **mouseutil** = require('@campfirehci/mouseutil')({
“fullscreen": true, // toggles fullscreen mode for windows
"display": true, // toggles browser displays for wrangler.
"screenWrap": true, // toggles screen wrapping for wall.

"Centermode": false, // Transition from wall-floor moves mouse to center
"floorURL": floorURL,
"wallURL": wallURL });

Implementation

a. **Mouse Controller** - JS object that stores the logic for mouse wrangler. Due to the nature of JavaScript event listeners, some functions had to be implemented in the local scope of the event listener functions.
   i. **[30-36]** function **init (args)**
      1. var **args** - An object of Key:Value pairs. These values are used throughout the program and initialized in the electron application.
         a. "FloorURL": String of weblink of page to display on floor.
            i. Ex: "FloorURL": "http://bit.ly/CampfireFloorSlide"
         b. "WallURL": String of weblink of page to display on wall.
            i. Ex: "WallURL": "http://bit.ly/CampfireFloorSlide"
         c. If using a file format the string like this:
            i. 'file://' + __dirname + 'image_name.png'
            ii. Files must be added to the npm directory (See **Section 3**)
   ii. **[59 - 80]** function **setScreens()**
      1. Initializes electron screen variable, iterates through available displays and finds widest screen to set as wall, second screen as floor.
   iii. **[81 - 136]** function **CreateWindow()**
      1. Creates browser windows for each screen using the electron BrowserWindow object. A weblink or file path can be provided to set the Floor/Wall screens to specific pages/images. Forces content size of browser window for each screen to corresponding screen sizes.
         a. Ex: BrowserWindow.loadURL(url)
            i. If url is a file: url should be a string with this format: 'file://' + __dirname + filename

ii. If url is a weblink, then it should be a string with the exact weblink: 'http://bit.ly/CampfireFloorSlide'

iv. [155 - 299] function **listener()**

1. Initializes variables for mouse movement event opener using **win-mouse.**
2. **[162]** var **OnFloor:** boolean used to check if the mouse is within the current bounds of the designated floor monitor.
3. **[165 - 200]** var **Util**: Object storing functions locally in the scope of the listen() function.
    a. **[170-176]** function **calcTheta(dx, dy)**
        i. Calculates an angle from the difference in position (**dx, dy**) between mouse and center of floor screen.
    b. **[177-181]** function **onFloor(x, y, fb)**:
        i. Takes current mouse pos **x** and **y** and checks if they are within the the floor bounds(**fb)**.
    c. **[183-200]** function **screenWrap(x, y, wb)**:
        i. Checks current position on wallscreen, if near vertical edge of screen, perform left-to-right or right-to-left mouse transition for appropriate scenarios.
4. **[200-227]** function **wallListener(x, y, fCx, fCy, fRadius, wallOffset, borderOffset)**
    a. The event listener uses this function when onFloor returns false. Converts current mouse **x** position into percentage over the total wall screen width. Converts the value into an angle. Checks when mouse moves near bottom of wall screen to transition to floor.
    b. fCy is the y position of the floor center
    c. fCx is the x positon of the floor center
    d. Offsets are used because if we consider angle 0 to be right of center from 12 o’ clock, then the wall origin does not match up with the floor origin. See the code for more details on how offsets are used.
5. **[229 - 257]** function **floorListener(x,y,fCx, fCy, fRadius, floorOffset, borderOffset)**

a. The event listener calls this function when the wrangler knows that the mouse cursor is within the bounds of the floor.
b. The wrangler computes the distance from the center to the current mouse position, obtains a theta value from this distance using atan and checks when the current radius of the mouse cursor is near the maximum radius of the floor screen to transition it to the wall floor.
c. **borderOffset** is the amount of distance before the edge of a screen when the wrangler transitions the mouse cursor to the other screen. I.e. for some fRadius = 540, 540 - (borderOffset = 2) = 538, when the user's mouse has reached a radius of 538 or greater, than the wrangler should transition the mouse.

Updating MouseWrangler on NPM

1. https://www.npmjs.com/package/@campfirehci/mouseutil/
2. **https://docs.npmjs.com/getting-started/publishing-npm-packages**
3. Open cmd.exe, npm login
4. cd to campfire-hci/npm
    a. Update code by modifying index.js
    b. Increment version number of package.json
    c. npm publish
5. Adding files to node package
    a. Add desired image to npm/images/ folder in project.
    b. Update url parameter to this file location.
        i. 'file://' + __dirname + "/images/" + filename
6. Conflicting version errors,
    a. Use npm version patch then npm publish.

Future Work

1. Find a way to accept user input from terminal or interface to mouse wrangler.
2. Mouse support for external screens as well as the campfire screens.
3. Rotate mouse cursor in direction of movement.
4. Create a larger toolkit that this code can reside and be executed in.
5. Allow flexibility for multiple display setups and coordinate systems.

Additional Notes

The best way to check if the screens are set up properly, is to console.log the allScreens variable found in index.js of the MouseWrangler folder. If app icon size is set at 100%, the scale factor will be 1 and the coordinate system should consist of the wall screen with origin (0, 0) and size (6400, 800) and floor screen with origin (-1920, 0) and size (1920, 1080)

```
all screens [ { id: 2528732444,
   bounds: { x: -1920, y: 0, width: 1920, height: 1080 },
   workArea: { x: -1920, y: 0, width: 1920, height: 1080 },
   size: { width: 1920, height: 1080 },
   workAreaSize: { width: 1920, height: 1080 },
   scaleFactor: 1,
   rotation: 0,
   touchSupport: 'unknown' },
 { id: 2779098405,
   bounds: { x: 0, y: 0, width: 6400, height: 800 },
   workArea: { x: 0, y: 0, width: 6400, height: 800 },
   size: { width: 6400, height: 800 },
   workAreaSize: { width: 6400, height: 800 },
   scaleFactor: 1,
   rotation: 0,
   touchSupport: 'unknown' } ]
```

Consider each problem separately, the floor to wall transition and wall to floor transition. The floor to wall transition is a mapping of the current theta obtained from the current mouse position to the x position of wall screen. The wall to floor is a reverse mapping of this.