Week 11 Tutorial - Phone Sensors!

In this tutorial, we will be using the phone tilt sensors as the interaction mechanism for our web content!

We will start from code (that you'll surely recognize) built on an array of animated bubbles. We'll serve the code so that we can use the sensors on mobile devices. A simple server is included so that we can serve from our PCs and use mobile devices as clients. (Some PCs also have tilt sensing though!)

Our goal looks something like this (before we add sensor interaction)**:**
https://nm2207.org/creativeweb/resources/emitterdemo_sm.mp4

First, an important word on Serving:

The https protocol is the **secure** version of http. When clients visit a securely served website, Information is exchanged between the server and the client to guarantee that the server is who they say they are.

It turns out that the motion (orientation) sensors on your phone can be used to infer your location, and even what buttons you press (when for example you enter a PIN #)! Thus, apps that use those sensors must be served securely.

nm2207.org runs a secure server for you by default (but if you run your own node.js server there, it is not secure and uses the regular http protocol). Thus if you wanted to upload today's challenges to nm2207.org, clients could use your app with their sensors just fine.

However, for development on your local machine you need to run a secure server in order to visit the page from your phone. Below are the instructions for installing and running a secure server on Windows and Macs:

First open a command line window (terminal) in the Challenge folder (which will have a www/ subfolder containing the index.html and the code as usual). Then:

Windows

```
> npm install -g https-localhost
> set PORT=8500
> serve www
> ipconfig
  (to get your ip address, for example 192.168.xxx.xxx)
```

Mac

```
> sudo npm install -g https-localhost
> sudo PORT=8500 serve www
> ifconfig | grep inet
  (to get your ip address, for example 192.168.xxx.xxx)
```

Now (on both platforms)

*Point computer browser to https://192.168.xxx.xxx:8500

You still may still get the 'Your connection is not private' warning

If so,

Click on Advanced

Click on Proceed to 192.168.xxx.xxxx (unsafe)

Challenges

Setup:

1. On your phone settings, set your Display "Auto-rotate screen" to "off" (so that when you tilt your phone, the display does not rotate)
2. In your editor,
    - make the disks small (eg 2-10 pixels)
    - check to make sure the code runs.
    - Get it running on your phone served from your local server. **Remember: your phone and computer have to be on the same LAN (connected to the same WIFI) network.**

Emit!

3. Make the midpoint of the paper an "emitter" that shoots out a disk every 100ms:
    - Hint: create a new interval timer with a function that "emits" by just setting the position of **one** disk in your array back to the center of the page (we did that for all disks when we initialized our array). Pass in an anonymous function to the timer where you will write the code to do the "emitting" (recentering) of a disk.
    - Hint: Each time you emit, pick a different disk in the array to move to the center. You'll also need a "next_to_emit" counter that you update in that function, so you know which disk is "next in line" to be recentered.
    - Hint: Use the "mod" operator, "%" to update the "next_to_emit" counter so that it doesn't reference disks beyond the length of the disk array. Remember, the mod operator gives you the remainder after dividing the first number by the second:
      0 % 10 // returns 0,
      6 % 10 // returns 6,
      10 % 10 // return 0,
      etc.
4. Add gravity:
    - Hint: You'll only need two lines of code, and one of them is just to create the gravity variable and initialize it to some value!
    - Hint (for the second line of code you need to add): **Use gravity to update yrate** in the draw() function in the same way we use our rate values to update the position values of the disks. (The only difference is that each disk has its own rate for updating position, where all disks will use the same gravity variable for updating their yrates.

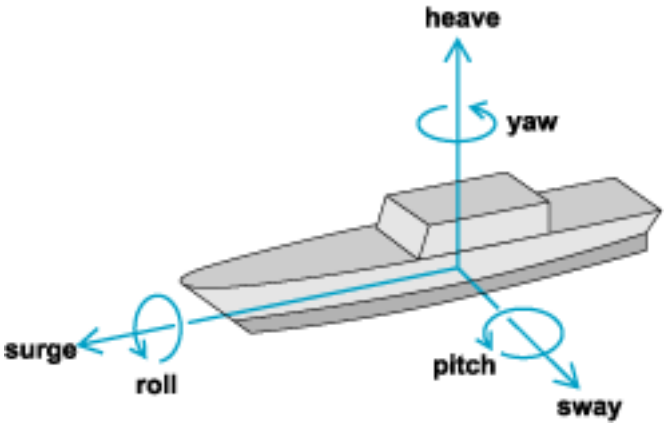
- Hint: you'll probably want to reinitialize the yrates where and when you recenter the disks. (Why? – think, discuss with your partner)

Next up: Device sensors

If window.DeviceOrientationEvent is defined, then you can receive 'deviceorientation' events on the 'window' object.(It is not defined for all devices, so you must check in your code).

If you listen for 'deviceorientation' events, **your callback function will be passed an object as an argument**. That object will have have properties:

- alpha - "yaw", in [0, 360] (this is the compass direction)
- beta - "pitch" in [-180,180] pointing down in -90, pointing up is 90
- gamma - "roll" in [-90,90]



SO:

5. if window.DeviceOrientationEvent is defined, then listen for the 'deviceorientation' events on the 'window' object, and set gravity using the beta property of the event your callback is passed. (You might need to alter beta to get it into a range usable for gravity).
6. print the gravity value to the aside div of the web page so you can see it change. Phones don't have the console window, so we are using the aside div for printing ourselves messages for info and debugging!
7. Make your disks bounce when they hit the top and bottom walls.
8. Now make your disks bounce back with reduced speed when they hit the walls (just as real balls bouncing lose energy with each bounce).

   Bonus round

9. Use touch-events to change something about how your app works. For example, you could make the position of the touch in the left-to-right direction cause a left-right gravity (or "wind") effect.

Comments

Notice how we are "using old disks" that have outlived their usefulness (gone off screed) rather than continuously producing new disks. This is a kind of "efficiency trick" that we have used before to manage the amount of computation required. A fixed number of disks means a fixed amount of computation for position updates and drawing.
You could make your code even more efficient if you avoided updating the position of the disks when they are off screen.

Different phones have different sensing capabilities. Some have "near field" sensors that know if you are close to them. Some phones have compasses and other many sensors that are also available for your interaction pleasure. Some phone games used video andour sound input for interaction. The possibilities are endless.

...