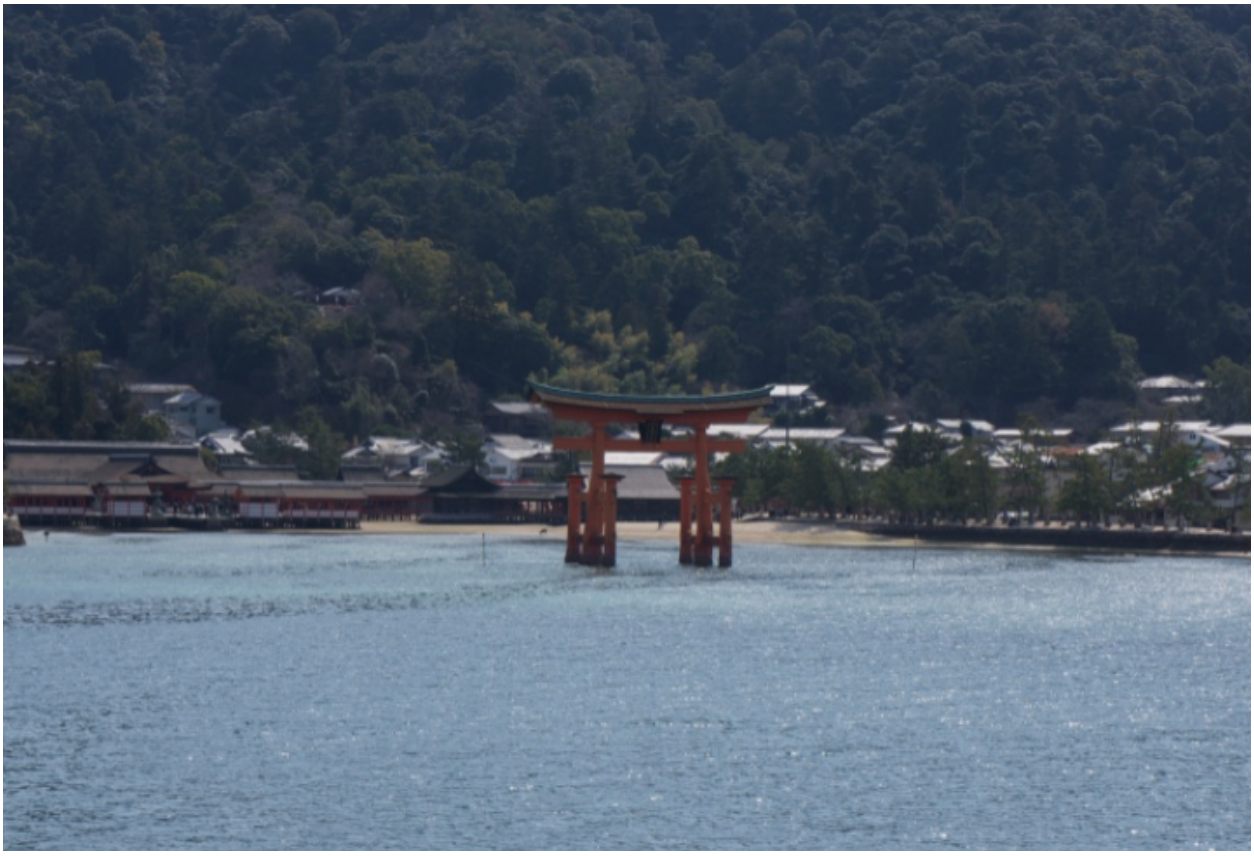
Net-centric introduction to computing

Pan & Tilt



m Jenkin + h Roumani

1. Introduction

Mobile devices — including the Android platforms used in EEC1012 — are typically equipped with a range of sensors related to their mobile nature. Some devices come equipped with sensors that allow the device to estimate its position with respect to satellites (GPS) so that the device can be localized over the surface of the Earth. Other sensors measure things like the direction to north, or the tilt of the device relative to gravity. How the platform actually measures these values is an interesting technical challenge. The sensors have to be inexpensive, low power and relatively accurate. They also have to deal with the fact that the actual device is subject to a range of accelerations related to the direction of gravity. If you choose, you can take courses in upper years that will consider how such sensors 'work' and how they can be integrated within a device such as a smart phone or tablet. Here, however, we will just take such sensors as 'working' and explore how web-based applications operating on a Tablet or Phone can access this information in the design of an application.

In this lab you will build what is perhaps the most straightforward lab that monitors the orientation of your device, a two dimensional level. In essence, you will update the visual display on the Android device so that it indicates the pitch and tilt of the device. There are many applications for the Android and IOS platforms that do this. In this lab you will write your own.

One reality of coding for the actual mobile device is that much debugging will not be possible except when you are running on the Android device itself. Much of what your application will do is tied to actually reading the value of the orientation sensor on the device. Your laptop, or the department's one, lacks such a sensor so testing/debugging on a laptop will have only limited functionality.

As with all labs in this course, this lab is laid out as an ePub. It is expected that you will have read the lab electronic book prior to attending the lab. In order to encourage this, each lab has an associated set of 'pre lab' assignments that must be completed prior to attending the lab. You will not receive a grade for this lab unless

you have completed the pre lab exercises. Exercises in the lab are to be documented in your ePortfolio lab book.

2. Background

This lab involves building an interface with HTML, formatting it using CSS, and implementing the logic through Javascript to monitor the orientation of the Android device relative to gravity. In this section we highlight the key features needed from each of these technologies.

You may want to skim through the background material in your first review of this lab. After wards, go to the actual exercises associated with the lab and then refer back to the material presented here as needed.

2.1 Manipulating onscreen positions using JavaScript

In earlier labs you manipulated HTML from JavaScript using document.getElementById("id") to find the element, and then using the innerHTML property to change the text associated with the element. It is also possible to change the CSS associated with the element. The style information is mentioned in the style property of the element. The style property is a Style object, and has a large number of properties that can be manipulated, properties that are directly related to the CSS properties associated with the element itself. The full list of such properties can be found here. For this lab, the critical style parameters are 'top' and 'left' (the locations of the upper left corner of an element). Properly manipulating these values will move the location of the element on the page. One final (but important property) for this lab is the z-index style (zIndex in JavaScript). This property defines the stack order of overlapping elements. Elements with a higher z-index value are drawn 'on top of' elements with a lower z-index value.

2.2 The orientation of a device in JavaScript

A major difference between traditional computer platforms (laptops, etc.) and portable devices (e.g., phones and tablets) is that traditional computer platforms have a natural orientation associated with them. They have a bottom and top if you like. Devices like tablets can be held in many different ways, even upside down. In order to accommodate this, many portable devices come equipped with sensors to monitor their orientation relative to two standard references on Earth: their orientation relative to gravity and their orientation relative to magnetic north.

Of course, not all devices come equipped with such sensors, and unfortunately not all those that do utilize exactly the same standards in terms of default orientation, or direction of signs. Worse, not all browsers implement the standards in the same way. That being said:

- If a device supports device orientation events then the value of window.DeviceOrientationEvent will be true. This does not mean that the device actually will generate such events, rather that the software infrastructure 'works'.
- Assuming that the software supports orientation events, it is possible to associate a handler or callback function that will be invoked every time that there is a change in the orientation of the device. This is accomplished using the addEventListener method of the window.

The following javascript provides a snippet of how this works in practice:

```
function init() {
  document.getElementById("count").innerHTML = 0;
  if(window.DeviceOrientationEvent) {
    document.getElementById("log").innerHTML = "Supports orientation events";
    window.addEventListener("deviceorientation", orientationCallback, true);
  } else {
    document.getElementById("log").innerHMTL = "No orientation event
support";
  }
}

function orientationCallback(eventData) {
  document.getElementById("count").innerHTML =
Number( document.getElementById("count").innerHTML) + 1;
  document.getElementById("tilt").innerHTML = eventData.gamma;
  document.getElementById("pitch").innerHTML = eventData.beta;
}
```

```
alert("javascript loaded");
window.onload = init;
```

The callback is provided an eventData object reference. This object contains many fields, but two are of interest here; gamma and beta. For an Android device in its 'native' mode, these are the tilt and pitch.

This introduces one final little twist. Not all devices have the same 'native' mode. Large tablets often have a native mode that is portrait, as do phones. Smaller tablets often have a native mode of 'landscape'. What this means to the programmer is that to make the software perform correctly, the roll and pitch values will have to be modified based on the current mode of the device. This can be accomplished taking into account the value of window.orientation, which tells you the rotation of the screen display relative to the native orientation of the device.

2.4 Creating and displaying images

In earlier labs you drew on the screen using graphical primitives. An alternative approach is to draw graphical regions (rectangles) using pixel maps (pixmaps) obtained from elsewhere. The html entity will render the image 'foo.gif'. Images are rectangular. They have a width and height and are positioned using their upper left corner. Images are stored in one of a number of different formats (jpeg, gif, others) and most browsers support a range of different image formats. Some image formats (e.g., gif) support semi-transparent images. In essence these images have a channel that encodes the transparency of the image. Often known as the 'alpha channel', this channel encodes, per pixel of the image, the opacity or transparency of that pixel. The ability of an image format to permit transparency information to be encoded allows rectangular images to appear circular or indeed of any shape.

There are many tools for creating and editing images. The CENTOS image provided in this course includes a copy of the 'gimp' image editing tool. You can use gimp to create images of almost anything (although it will not make an artist

out of someone who has little artistic ability.). Gimp is free, and available for a wide range of platforms. By default, gimp does not save images in gif format, but it will import and export images in gif format. Full details on gimp can be found on line.

3. Exercises

As with other labs in this course, lab exercises are broken down into three sections, A, B and C. Exercises in section A are Pre-Lab exercises. All Pre-lab exercises **must** be completed prior to attending your lab. You will not be allowed to participate in the lab if you have not completed these exercised prior to attending the lab. Furthermore, you will not receive course credit for a lab for which the Part A lab was not completed. You will also get much more out of each laboratory if you spend some time going through the B and C exercises for each laboratory before attending your laboratory session.

A. Pre-lab

1. Download eecs1012pan.zip from the jr web page and unzip it. This is a simple web page that monitors the orientation of the device and updates a text display based on this. Examine them html and JavaScript provided.
2. Complete the pre-lab quiz on the course moodle page.

B. In-lab

1. Connect the Android device to the laptop and power up the CENTOS virtual image. Download eecs1012pan.zip to the server/PAN directory on the virtual machine and unpack it. Then load the web site into the html-apk application on the Android device. (To do this, open the html-apk application and enter the URL http://ip:8000/zip/PAN where ip is the ip address of your laptop. Run the application on the device. What is the range of pan and tilt reported by the device? Which direction is positive or negative? What happens when you rotate the device by more than 90 degrees from landscape to portrait and back again?
2. Use the settings tab on the application to lock the device into portrait (or landscape mode) and re-start the application. How does this change the definitions of roll and pitch? What is the natural orientation of your device?

3. Using gimp create square graphical objects. One will be used to mark the centre of the screen (call this cross.gif). The other will be used to mark the level. Call this ball.gif. Make them roughly 64 x 64 pixels in size. The level should be circular with the non-circular portion of the image transparent.
4. Rename rollaball.js to rollaball.js.old. Then, using atom, edit index.html so that it contains a link to a style sheet called rollaball.css and a link to a JavaScript script called rollaball.js. The body of index.html should consist of a with id "screen". Inside this place three , one with id "text" containing "this is some text", one with id "ball" containing the <img> tag linked to your ball.gif, and one with id "center" containing the <img> tag linked to your cross.gif. Test this web page using the previewer in atom. What is displayed. Describe this process in your ePortfolio using screen dumps or video capture.
5. Using atom edit rollaball.css and define the following styles.
    5.1. Define a style that sets padding, margin, border and outline of all <body> tags to the value 0. This will cause the body to extend all the ways to the boundary of the window (or screen) on which it is displayed.
    5.2. Define a style that impacts all tags with id "screen". (You will observe that you created a tag with this property above.) This style should set the background-color to be yellow, and the width and height to be 100%. The style should also set the position property to the value "relative". The importance of this last style property-value pair will be made clear in a moment. Run the web page in the atom previewer. Run it using Firefox and run it on the Android device. What do you see now? Document this in your ePortfolio.
    5.3. Define a style that impacts all tags with id "ball". Set the property of left to be 100px, the property of top to be 0px, the property of position to be absolute and the property of z-index to be 2.
    5.4. Define a style that impacts all tags with id "center". Set the property of left to be 0px, the property of top to be 0px, the property of position to be absolute and the property of z-index to be 1.

5.5. Define a style that impacts all tags with id "text". Set the property of left to be 0px, the property of top to be 0px, the property of position to be absolute, the property of z-index to be 1000, and the property of color to be black. Re-run the application in Firefox and on the Android device. Where do your shapes appear? Where does the "text" appear? What happens if you set the top property of "ball" to be 100px? Again, document all of this in your ePortfolio

6. Now rename rollaball.js.old to rollaball.js end edit it using atom. Earlier you demonstrated that this code successfully monitors the orientation of the device. You will now edit it so that the location of the image "ball.gif" will be updated based on the orientation of the device. Do not run the application now as it will crash — the id's of the html have changed.

6.1. In the init function code, rather than using the "log" id, use the "text" id and delete the line that sets the innerHTML for the "count" id to zero. Then delete the three lines in the body of the orientationCallback function. This should result in a workable (although mostly empty) JavaScript script. Run this using Firefox (with firebug enabled to catch any JavaScript errors you may introduce). What do you expect to see when you run this code on Firefox? Once the code is 'working', load it onto the Android device and start it. What do you expect to see when this code is run?

6.2. In the orientation callback, set the innerHTML of the text div to the value of the eventData.gamma followed by a blank followed by eventData.beta. This will update the status line showing the current roll and pitch of the device. Test this on the Android device to verify that this is working.

6.3. At the end of the orientationCallback create local variables tilt and pitch. Set tilt to be eventData.gamma and pitch to be eventData.beta. Once this is done, clamp pitch and tilt to the range -90..90. To do this, examine the value of pitch. If it is less than -90 set it to -90. If it is greater than 90 then set it to 90. Repeat this for pitch. This clamping is necessary because otherwise you will get roll and pitch values in the range -180 to 180 degrees.

6.4. You are now going to update the location of the centre marker in the orientationCallback function. Get the element whose id is "center" in the

variable d. The window in which you are drawing has an inner width in the field window.innerWidth and an inner height in the field window.innerHeight. The "center" element is an image. Its width is in d.offsetWidth and its height is in d.offsetHeight. Create a var x0 whose value is (window.innerWidth - d.offsetWidth) / 2 and a var y0 whose value is (window.innerHeight - d.offsetHeight) / 2. This is the (x,y) location of the upper left hand corner of the required location of this object, so set d.style.top to be y0 and d.style.left to be x0.

6.5. Finally you are now going to update the location of the tilt marker in the orientationCallback function. Get the element whose id is "ball" in the variable c. Create a var x whose value is (window.innerWidth - d.offsetWidth) / 2 and a var y whose value is (window.innerHeight - d.offsetHeight) / 2. As before, this is the centre location on the screen. This is also the amount of white space above and to the left of the centre of the screen. Now create a variable dx whose value is x * tilt / 90 and a variable dy whose value is y * pitch/90. Set the top of the "ball" to be y + dy and the left location of the "ball" to be x + dx. It should now move with roll and pitch. Document and demonstrate this system running in your ePortfolio.

7. In order to obtain credit for this lab you must demonstrate your ePortfolio to your TA. You may find it easier to document the operation of your application using a video captured from some other camera showing changes in the display with motion of the device.

C. Advanced

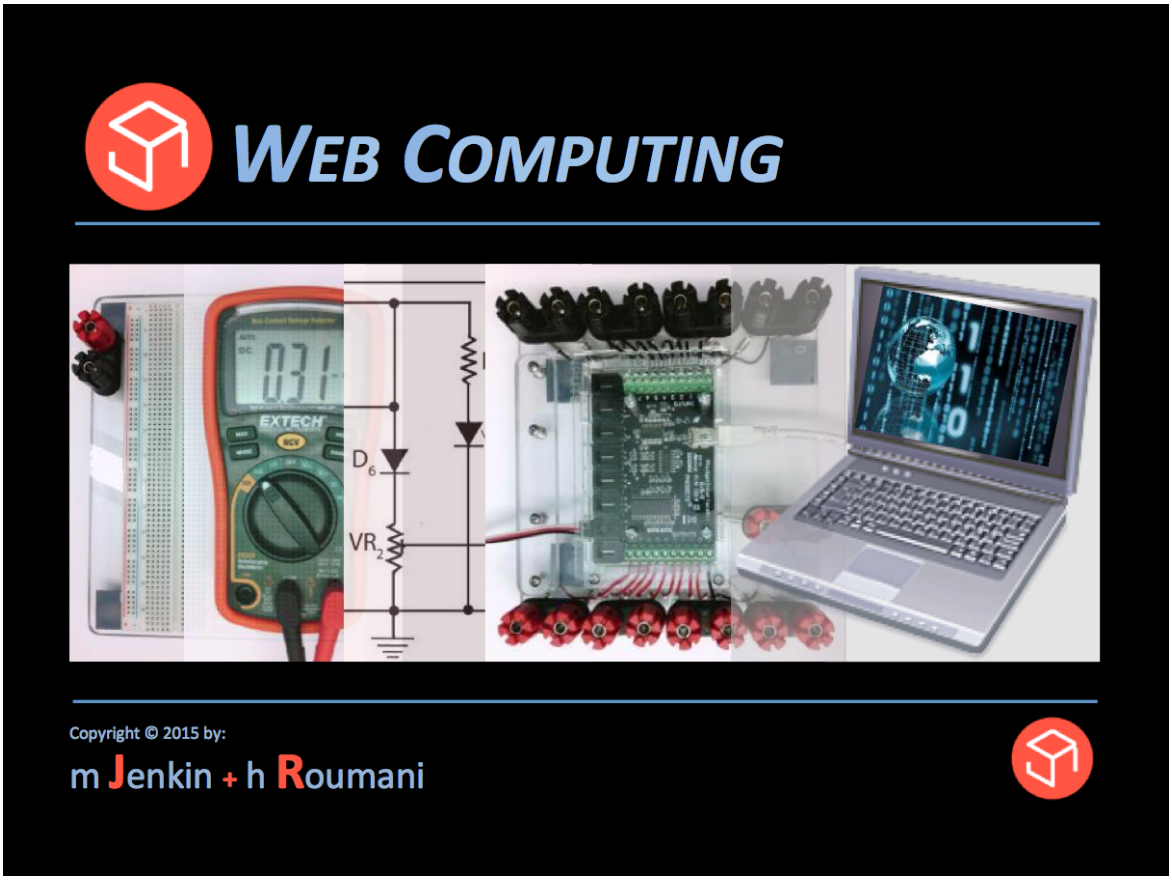
1. There are many different ways you can augment the basic lab. Perhaps the easiest is to make the markers much more attractive than those you generated first. Make the centre mark a cross. (It does not have to be the same size as the ball moving around). Make the ball look like a bubble. Change the background colour to be more appealing.
2. The orientation marker does not move like a bubble in a level. In fact, it goes the other way. Change the code so that the bubble moves like a bubble in a level.

3. As written in the basic lab, the code only works if the screen is locked into the ‘native’ orientation of the device. The value of window.orientation will tell you the rotation of the device relative to its native orientation. Use this value to correct the pitch and roll so that they are rotated when the device is rotated relative to the native orientation of the device.

4. Further Reading

- Full details on gimp can be found on line.

5. Credits


WEB COMPUTING
EXTECH
D6
VR2
Copyright © 2015 by:
m Jenkin + h Roumani