

IT452 Advanced Web and Internet Systems

Fall 2009

Set 6: Events and Animation

Animation and Events

- Animation – attention, explanation
 - Capturing user events
 - Education and games
-
- We focus particularly on techniques with YUI

Preliminaries

```
<style type="text/css">
  /*margin and padding on body element
   can introduce errors in determining
   element position and are not recommended;
   we turn them off as a foundation for YUI
   CSS treatments. */
  body {
    margin:0;
    padding:0;
  }
  #foo {width:10px; height:10px;background-color:#00f;}
</style>

<link rel="stylesheet" type="text/css"
 href="http://www.cs.usna.edu/~lmcdowell/courses/it452/yui/fonts/fonts-
min.css?_yuiversion=2.3.1" />
```

YUI-loader – modules needed:

```
require: ["event", "dom"],
```

falling.html – An event loop

```
// This function is periodically called by the timer.  It makes things move
function mainLoop() {
  moveBlock();
}
// (x,y) coordinates of the block
var blockPos = [200,200];

// Move the block by a little bit
function moveBlock() {
  blockPos[0] = blockPos[0] + 1;
  blockPos[1] = blockPos[1] + 2;
  YAHOO.util.Dom.setXY('foo', blockPos);
}

// Do necessary setup on first load
function init() {
  // Call main loop every 50 milliseconds
  setInterval("mainLoop()", 50);

  // On every click, call the move() function
  YAHOO.util.Event.on(document, "click", moveOnClick);

}
// When we click, move box to that location
function moveOnClick(e) {
  clickLoc = YAHOO.util.Event.getXY(e);
  blockPos[0] = clickLoc[0];
  blockPos[1] = clickLoc[1];

  // This command would immediately move the block to the click point
  //      YAHOO.util.Dom.setXY('foo', YAHOO.util.Event.getXY(e));
}
```

falling.html (HTML portion)

```
<body class="yui-skin-sam" >  
  
<h1>Using an event loop</h1>  
  
<!-- The actual block that moves around -->  
<div id="foo"></div>  
  
<div style="width:500px; height:200px; background-color:red">  
</div>
```

YUI Event Listeners

```
We used  
YAHOO.util.Event.on(document, "click", move);  
This is same as  
YAHOO.util.Event.addListener(document, "click", move);  
Can take optional arguments  
Boolean addListener ( el , sType , fn , obj , override )  
obj - object to be passed as an argument to the handler  
override - affects scope, see API  
  
sType takes many values - depends on browser. Some  
onfocus  
onblur  
keydown, keypress  
select
```

See <http://www.quirksmode.org/dom/events/index.html>

bounce.html (1)

```
var bigdivPos = [100,100];
var bigdivSize = [500,200]

// Move the block by a little bit
function moveBlock() {
    blockPos[0] = blockPos[0] + blockVeloc[0];
    blockPos[1] = blockPos[1] + blockVeloc[1];
    YAHOO.util.Dom.setXY('foo', blockPos);

    keepItemInsideBox(bigdivPos, bigdivSize, blockPos, blockSize, blockVeloc);
}

// Do necessary setup on first load
function init() {
    // Call main loop every 50 milliseconds
    setInterval("mainLoop()", 50);

    // On every click, call the move() function
    YAHOO.util.Event.on(document, "click", moveOnClick);

    // Move the big div element to a known location
    YAHOO.util.Dom.setXY('bigdiv', bigdivPos);
}
```

bounce.html (2)

```
// Checks to see if the block has hit the edge of the bigdiv.  If so, make it bounce
off
// If we get outside the box, this will also force us to move back in.
function keepItemInsideBox(boxPos, boxSize, itemPos, itemSize, itemVeloc) {

    // Check to see if we hit a vertical edge
    var edgeHit = calcVerticalEdgeHit(boxPos, boxSize, itemPos, itemSize);
    if ( (edgeHit == HIT_LEFT) && (itemVeloc[0] < 0) ) {
        // if too far left, make sure we go right
        itemVeloc[0] *= -1;
    }
    if ( (edgeHit == HIT_RIGHT) && (itemVeloc[0] > 0) ) {
        // if too far right, make sure we go left
        itemVeloc[0] *= -1;
    }

    // Check to see if we hit a horizontal edge
    var edgeHit = calcHorizontalEdgeHit(bigdivPos, bigdivSize, blockPos, blockSize);
    if ( (edgeHit == HIT_TOP) && (itemVeloc[1] < 0) ) {
        // if too far up, make sure we go down
        itemVeloc[1] *= -1;
    }
    if ( (edgeHit == HIT_BOTTOM) && (itemVeloc[1] > 0) ) {
        // if too far down, make sure we go up
        itemVeloc[1] *= -1;
    }
}
```

bounce.html (3)

```
// See if the item hit a vertical edge by going too far left or right.  
// NOTE: we assume it was basically inside the box to begin with  
function calcVerticalEdgeHit(boxPos, boxSize, itemPos, itemSize) {  
    // check for hit on left side  
    var boxLeft = boxPos[0];  
    var boxRight = boxPos[0] + boxSize[0];  
    var itemLeft = itemPos[0];  
    var itemRight = itemPos[0] + itemSize[0];  
  
    if (itemLeft < boxLeft)  
        return HIT_LEFT;  
    else if (itemRight > boxRight)  
        return HIT_RIGHT;  
}  
  
// See if the item hit a horizontal edge by going too far up or down.  
// NOTE: we assume it was basically inside the box to begin with  
function calcHorizontalEdgeHit(boxPos, boxSize, itemPos, itemSize) {  
    // check for hit on left side  
    var boxTop = boxPos[1];  
    var boxBot = boxPos[1] + boxSize[1];  
    var itemTop = itemPos[1];  
    var itemBot = itemPos[1] + itemSize[1];  
  
    if (itemTop < boxTop)  
        return HIT_TOP;  
    else if (itemBot > boxBot)  
        return HIT_BOTTOM;  
}
```

Multiblocks.html

```
// (x,y) coordinates of the block  
var block1Pos = [200,200];  
var block1Size = [10,10]  
var block1Veloc = [1, 2];  
  
var block2Pos = [300,300];  
var block2Size = [10,10]  
var block2Veloc = [2, -1];  
  
function mainLoop() {  
    moveBlock('block1', block1Pos, block1Size, block1Veloc);  
    moveBlock('block2', block2Pos, block2Size, block2Veloc);  
}  
  
// Move the block by a little bit  
function moveBlock(name, pos, size, veloc) {  
    pos[0] = pos[0] + veloc[0];  
    pos[1] = pos[1] + veloc[1];  
    YAHOO.util.Dom.setXY(name, pos);  
  
    keepItemInsideBox(bigdivPos, bigdivSize, pos, size, veloc);  
}  
..  
<body class="yui-skin-sam" >  
  
<h1>Using an event loop</h1>  
  
<!-- The actual block that moves around -->  
<div id="block1" class="myblock"></div>  
<div id="block2" class="myblock"></div>
```

Exercise

1. How would you make the blocks twice as FAST?
2. How would you make the blocks twice as BIG?
3. What is a second way to make the blocks move TWICE as fast?

Game.html (1)

```
// missile launcher
var launcherPos = [350,300];
var launcherSize = [ 20, 20]

// a missile. Put off board initially
var offBoardPos = [800, 0];
var missilePos = offBoardPos;
var missileSize = [50,50];
var missileVeloc = null;

// Do necessary setup on first load
function init() {
    // Call main loop every X milliseconds
    setInterval("mainLoop()", 25);

    // On every click, call the move() function
    YAHOO.util.Event.on(document, "click", fireAtClick);

    // Move the big div element to a known location
    YAHOO.util.Dom.setXY('bigdiv', bigdivPos);

    // Create the blocks
    createDiv(block1Pos, block1Size, "myblock", "block1");
    createDiv(block2Pos, block2Size, "myblock", "block2");

    // Create the missile launcher
    createDiv(launcherPos, launcherSize, "launcher", "launcher");

    // missile initially invisible
    missileDom = createDiv (missilePos, missileSize, "invisible", "missile1");
}

}
```

Game.html (2)

```
// Create a div with the given pos, size, and class
function createDiv(pos, size, myclass, myid) {
    // create basic div
    var aDiv = document.createElement("div");
    aDiv.className = myclass;
    aDiv.id = myid;
    aDiv.style.width = "+"+size[0]+"px";
    aDiv.style.height = "+"+size[1]+"px";
    document.body.appendChild(aDiv);

    // move it to right place
    YAHOO.util.Dom.setXY(aDiv, pos);

    return aDiv;
}

function incrementScore(val) {
    score = score + val;
    document.getElementById('scorebox').innerHTML = "Score: "+score;
}
```

Game.html(3)

```
// Check for a missile hit. If hit, act like block destroyed, get rid of missile
function checkMissileHit(missileId, missilePos, missileSize, blockPos, blockSize) {
    if (!didSpheresHit(missilePos, missileSize, blockPos, blockSize)) {
        return;
    }

    // we did have a hit. Make the missile invisible and immobile
    document.getElementById("missile1").className = "invisible";
    missileVeloc = null;
    missilePos = offBoardPos;
    YAHOO.util.Dom.setXY(missileId, missilePos);
    incrementScore(100);

    // Randomize location of new block (somewhere inside bigdiv)
    blockPos[0] = bigdivPos[0] + Math.random() * bigdivSize[0];
    blockPos[1] = bigdivPos[1] + Math.random() * bigdivSize[1];
}

// Check if the spheres with given pos and sizes are touching.
// Assumes width equal to height
function didSpheresHit(pos1, size1, pos2, size2) {
    // We have positions of top left corners. Convert to get center positions
    var cx1 = pos1[0] + size1[0] / 2;
    var cx2 = pos2[0] + size2[0] / 2;
    var cyl = pos1[1] + size1[1] / 2;
    var cy2 = pos2[1] + size2[1] / 2;

    // find distance between centers
    var dx = cx1 - cx2;
    var dy = cyl - cy2;
    dist = Math.sqrt(dx*dx + dy*dy);

    // size tracks the width, so radius would be half of that.
    // So they touch if dist less than (width1 + width2) / 2
    if (dist < ( (size1[0] + size2[0]) / 2) ) {
        return true;
    }
    return false;
}
```

Game.html (4)

```
// When we click, fire a missile at that location
function fireAtClick(e) {
    clickLoc = YAHOO.util.Event.getXY(e);

    // calculate slope toward that spot
    var dx = clickLoc[0] - launcherPos[0];
    var dy = clickLoc[1] - launcherPos[1];

    // scale down by the overall distance.  not necc. the right math, but
    // gets it moving in the right direction
    var dist = Math.sqrt(dx*dx + dy*dy);
    dx = 2 * dx / dist;
    dy = 2 * dy / dist;

    // Make coordinates for the missile
    missileVeloc = [ dx , dy ];
    missilePos = [ launcherPos[0], launcherPos[1] ]; // make sure get COPY of that
array, not same one

    // Missile already created, but maybe invisible - this will make it visible.
    missileDom.className = "missile";
}
```

Game Design

- Think about the key elements
 - Mechanics
 - Story
 - Aesthetics
 - Technology
- JS/YUI speed may be an issue
 - Test early and often!
 - Design and test
- Start thinking about your game!