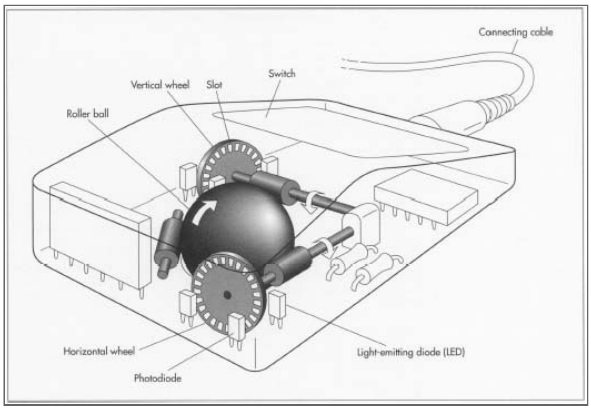
Dealing with mouse events

REMINDERS FROM THE HTML5 PART 1 COURSE



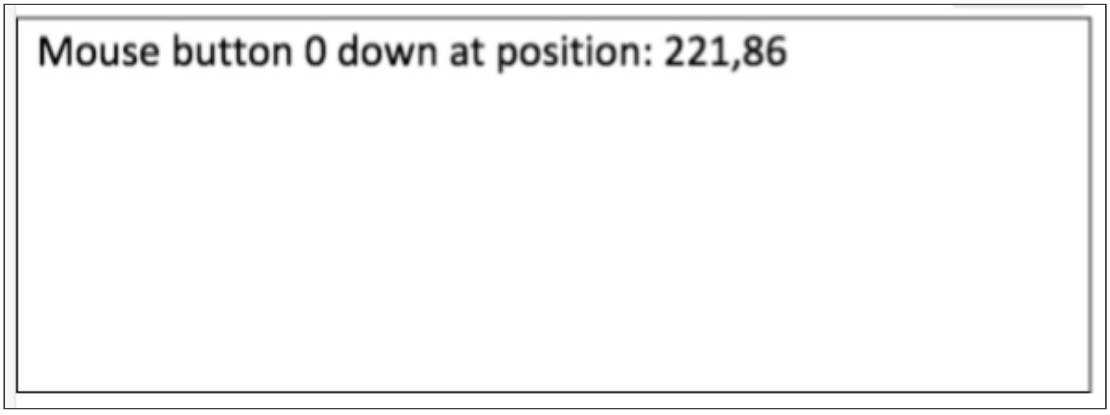
Working with mouse events means detecting when the mouse button is up or down, identifying the button, keeping track of mouse moves, getting the x and y coordinates of the cursor, etc.

Special care must be taken when getting the mouse coordinates as the HTML5 canvas often has default CSS properties that would produce false coordinates. The trick to get the right x and y mouse cursor coordinates is to use this method from the canvas API:

```
// necessary to take into account CSS boudaries
var rect = canvas.getBoundingClientRect();
```

The width and height of the `rect` object must be taken into account. These dimensions correspond to the padding / margins / borders of the canvas. See how we deal with them in the `getMousePos()` function in the next example.

Here is an online example at JSBin that covers all cases correctly.



Just move the mouse over the canvas and press or release mouse buttons. Notice that we keep the state of the mouse (position, buttons down or up) in the `inputStates` object, like we did with the keys in the previous lesson.

Below is the JavaScript source code for this small example:

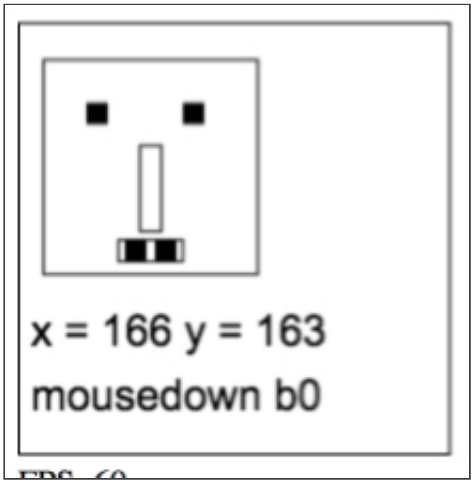
```
var canvas, ctx;
var inputStates = {};
window.onload = function init() {
   canvas =document.getElementById('myCanvas');
   ctx = canvas.getContext('2d');
   canvas.addEventListener('mousemove',function (evt) {
      inputStates.mousePos =getMousePos(canvas, evt);
      var message = 'Mouse position:
' +inputStates.mousePos.x + ',' +
        inputStates.mousePos.y;
      writeMessage(canvas, message);
   }, false);
   canvas.addEventListener('mousedown',function (evt) {
      inputStates.mousedown = true;
      inputStates.mouseButton = evt.button;
      var message = "Mouse button " +evt.button + " down at
position: " +

inputStates.mousePos.x +',' + inputStates.mousePos.y;
      writeMessage(canvas, message);
   }, false);

   canvas.addEventListener('mouseup',function (evt) {
      inputStates.mousedown = false;
      var message = "Mouse up at position:
"+ inputStates.mousePos.x + ',' +
                    inputStates.mousePos.y;
      writeMessage(canvas, message);
   }, false);
};
function writeMessage(canvas, message) {
  var ctx = canvas.getContext('2d');
```

```
    ctx.save();
    ctx.clearRect(0, 0, canvas.width,canvas.height);
    ctx.font = '18pt Calibri';
    ctx.fillStyle = 'black';
    ctx.fillText(message, 10, 25);
    ctx.restore();
}
function getMousePos(canvas, evt) {
    // necessary to take into account CSS boudaries
    var rect = canvas.getBoundingClientRect();
    return {
        x: evt.clientX - rect.left,
        y: evt.clientY - rect.top
    };
}
```

INCLUDE THE MOUSE LISTENERS INTO THE GAME FRAMEWORK

Now we will include these listeners into our game framework. Notice that we changed some parameters (no need to pass the canvas as a parameter of the `getMousePos()` function, for example).

The new online version of the game engine can be tried at JSBin:



Try pressing arrows and space keys, move the mouse and press the buttons, all at the same time. You'll see that the game frameworks handle all these events simultaneously,

as the keyboard/mouse state is stored in a global variable named `inputStates`, that is checked every 1/60th second and updated on key/mouse events.

JavaScript source code:

```
// Inits
window.onload = function init() {
  var game = new GF();
  game.start();
};
// GAME FRAMEWORK STARTS HERE
var GF = function(){
  ...
  // Vars for handling inputs
  var inputStates = {};
  var measureFPS = function(newTime){
      ...
  };
  // Clears the canvas content
  function clearCanvas() {
    ctx.clearRect(0, 0, w, h);
  }
  // Functions for drawing the monster and perhaps other
objects
  function drawMyMonster(x, y) {
    ...
  }
  var mainLoop = function(time){
   // Main function, called each frame
   measureFPS(time);
   // Clears the canvas
   clearCanvas();
   // Draws the monster
   drawMyMonster(10+Math.random()*10,10+Math.random()*10);
   // Checks inputStates
   if (inputStates.left) {
     ctx.fillText("left", 150, 20);
   }
```

```
    if (inputStates.up) {
      ctx.fillText("up", 150, 40);
    }
    if (inputStates.right) {
      ctx.fillText("right", 150, 60);
    }
    if (inputStates.down) {
      ctx.fillText("down", 150, 80);
    }
    if (inputStates.space) {
      ctx.fillText("space bar", 140, 100);
    }
    if (inputStates.mousePos) {
      ctx.fillText("x = " +inputStates.mousePos.x + " y = " +
                              inputStates.mousePos.y, 5, 150);
    }
    if (inputStates.mousedown) {
      ctx.fillText("mousedown
b" +inputStates.mouseButton, 5, 180);
    }

    // Calls the animation loop every 1/60th of second
    requestAnimationFrame(mainLoop);
  };
  function getMousePos(evt) {
    // Necessary to take into account CSS boudaries
    var rect =canvas.getBoundingClientRect();
    return {
      x: evt.clientX - rect.left,
      y: evt.clientY - rect.top
    };
  }
  var start = function(){
    ...
    // Adds the listener to the main window object, and
updates the states
    window.addEventListener('keydown',function(event){
      if (event.keyCode === 37) {
        inputStates.left = true;
      } else if (event.keyCode === 38) {
```

```
        inputStates.up = true;
      } else if (event.keyCode === 39) {
        inputStates.right = true;
      } else if (event.keyCode === 40) {
        inputStates.down = true;
      } else if (event.keyCode === 32) {
        inputStates.space = true;
      }
    }, false);
    // If the key is released, changes the states object
  window.addEventListener('keyup',function(event){
     if (event.keyCode === 37) {
       inputStates.left = false;
     } else if (event.keyCode === 38) {
      inputStates.up = false;
     } else if (event.keyCode === 39) {
      inputStates.right = false;
     } else if (event.keyCode === 40) {
      inputStates.down = false;
     } else if (event.keyCode === 32) {
       inputStates.space = false;
     }
  }, false);
  // Mouse event listeners
  canvas.addEventListener('mousemove',function (evt) {
    inputStates.mousePos = getMousePos(evt);
  }, false);
  canvas.addEventListener('mousedown',function (evt) {
    inputStates.mousedown = true;
    inputStates.mouseButton = evt.button;
  }, false);
  canvas.addEventListener('mouseup',function (evt) {
    inputStates.mousedown = false;
  }, false);

  // Starts the animation
  requestAnimationFrame(mainLoop);
 };
 // Our GameFramework returns a public API visible from
outside its scope
```

```
    return {
      start: start
    };
};
```