Dealing with key events

REMINDERS FROM THE HTML5 PART 1 COURSE

This has been something of a nightmare for years, as different browsers had different ways of handling key events and key codes (read this if you are fond of JavaScript archaeology). Fortunately, it's much better today and we can rely on methods that should work on any browser that is less than four years old.

When you listen to keyboard related events (`keydown` or `keyup`), the event function will contain the code of the key that fired the event. Then it is possible to test what key has been pressed or released, like this:

```
window.addEventListener('keydown',function(event) {
    if (event.keyCode === 37) {
      // Left arrow was pressed
    }
}, false);
```

At *lines 2, 37* is the key code that corresponds to the left arrow. Below is a quick reminder of the key codes.

You can try key codes with this interactive example, and here is a list of keyCodes (taken from this Web site):

Key	Code
backspace	8
tab	9
enter	13
shift	16
ctrl	17
alt	18
pause/break	19
caps lock	20
escape	27
(space)	32
page up	33
page down	34
end	35
home	36
left arrow	37
up arrow	38
right arrow	39
down arrow	40
insert	45
delete	46
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
a	65
b	66
c	67
d	68

Key	Code
e	69
f	70
g	71
h	72
i	73
j	74
k	75
l	76
m	77
n	78
o	79
p	80
q	81
r	82
s	83
t	84
u	85
v	86
w	87
x	88
y	89
z	90
left window key	91
right window key	92
select key	93
numpad 0	96
numpad 1	97
numpad 2	98
numpad 3	99
numpad 4	100
numpad 5	101
numpad 6	102
numpad 7	103

Key	Code
numpad 8	104
numpad 9	105
multiply	106
add	107
subtract	109
decimal point	110
divide	111
f1	112
f2	113
f3	114
f4	115
f5	116
f6	117
f7	118
f8	119
f9	120
f10	121
f11	122
f12	123
num lock	144
scroll lock	145
semi-colon	186
equal sign	187
comma	188
dash	189
period	190
forward slash	191
grave accent	192
open bracket	219
back slash	220
close braket	221
single quote	222

GAME REQUIREMENTS: MANAGING MULTIPLE `KEYPRESS` / `KEYRELEASE` EVENTS

In a game, we often need to check which keys are down at a very high frequency, typically from inside the game loop, that is running up to 60 times per second.

If a spaceship is moving left, chances are you are keeping the left arrow down, and if it's firing missiles at the same time you must also be pressing the space bar like a maniac and maybe pressing the shift key to release smart bombs.

Sometimes these three keys might be down at the same time, and the game loop will have to take these three keys into account: move the ship left, release a new missile if the previous one is out of the screen or if it reached a target, launch a smart bomb if conditions are OK, etc.

Keep in a JavaScript object the list of keys that are pressed at any moment

The typical method is: store in an object the list of the keys (or mouse button or whatever game pad button...) that are up or down at a given time. For our small game engine we will call this object "`inputStates`".

We will update its content inside the different input event listeners, and we will check its value from inside the game loop, 60 times per second.

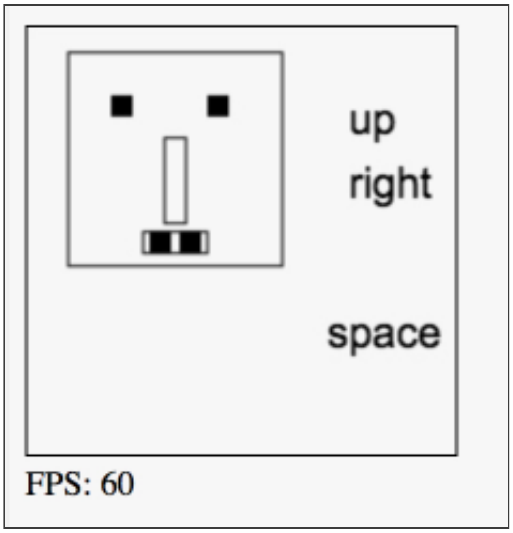
Add this to our game framework:

So, this is what we changed in our small game engine prototype (which is far from finished yet):

1. We added an empty `inputStates` object as a global property of the game engine,
2. In the `start()` method, we added the event listeners for the`keydown` and `keyup` events. In each listener, we tested if the arrow keys or the space bar have been pressed or released, and we set different properties of the inputStates object accordingly.
   For example, if the space bar is pressed, we do an`inputStates.space=true`; if it's released, we do an`inputStates.space=false`.

3. In the main loop, we added tests to check what keys are down; if one key is down, we draw its name in the canvas.

Here is the online example you can try at JSBin



And here is the complete 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);

     // check inputStates
     if (inputStates.left) {
       ctx.fillText("left", 150, 20);
     }
     if (inputStates.up) {
       ctx.fillText("up", 150, 50);
     }
    if (inputStates.right) {
       ctx.fillText("right", 150, 80);
    }
    if (inputStates.down) {
      ctx.fillText("down", 150, 120);
    }
    if (inputStates.space) {
      ctx.fillText("space bar", 140, 150);
    }
    // Calls the animation loop every 1/60th of second
    requestAnimationFrame(mainLoop);
  };
  var start = function(){
     ...
     // Important, we will draw with this object
     ctx = canvas.getContext('2d');
     // Default police for text
     ctx.font="20px Arial";
     // Add the listener to the main, window object, and
update 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, change 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);
   // Starts the animation
   requestAnimationFrame(mainLoop);
 };
 // our GameFramework returns a public API visible from
outside its scope
 return {
   start: start
 };
};
```

You may notice that on some computers / operating systems, it is not possible to simultaneously press the up and down arrow keys, or left and right arrow keys, as they are mutually exclusive. However space + up + right are OK.