UDACITY

Back to Front-End Web Developer Nanodegree

Classic Arcade Game Clone

REVIEW

CODE REVIEW 6

HISTORY

js/app.js 5

```
/*
* Initialize variables to help with movements:
* rowHeight and colWidth give me the step lengths to be centered
  in the next block (row or column)
* rowZero and colZero help me identify where the first row (from the top)
  and the first column begin so I have a sense of my origin, adjusted for
  character/sprite height and width.
*/
var rowHeight = 83; //height of rows
var colWidth = 101; //width of columns
var yOffsetForChar = -23; // Used to center the player/enemies in their row
var enemyRows = [
    // Array to store the rows for enemies to follow using the set rowHeight a
    // offset to help center the enemies in the rows
    yOffsetForChar + rowHeight,
    yOffsetForChar + rowHeight * 2,
    yOffsetForChar + rowHeight * 3
];
var enemySpeeds = [
    // Array to store the three speed levels for enemies, underlying function
    // getRandomInt uses Math.random for assured randomness
    getRandomInt(getRandomInt(1, 3), getRandomInt(4, 6)),
    getRandomInt(getRandomInt(4, 6), getRandomInt(8, 10)),
    getRandomInt(getRandomInt(8, 10), getRandomInt(15, 20))
];


// Thanks to MDN documentation for getRandomInt(),
```

```
29 // returns a random integer between min & max,
30 // inclusive/non-inclusive respectively.
31 function getRandomInt(min, max) {
32     /* Thanks to MDN documentation for getRandomInt(),
33     returns a random integer between min & max,
34     inclusive/non-inclusive respectively. */
35     min = Math.ceil(min);
36     max = Math.floor(max);
37     return Math.floor(Math.random() * (max - min)) + min;
38 }
39
40 /****************/
41 /* ENEMY CLASS */
42 /***************/
43 var Enemy = function() {
```

SUGGESTION

Great work declaring the Enemy class. You can ditch the usage of var and use let or const instead.

Check Udacity JS styling guide:

- http://udacity.github.io/frontend-nanodegree-styleguide/javascript.html

Check the difference between var, let and const:

- https://dev.to/sarah_chima/var-let-and-const--whats-the-difference-69e

```
44     // Variables applied to each of our instances go here,
45     // we've provided one for you to get started
46     // The image/sprite for our enemies, this uses
47     // a helper we've provided to easily load images
48     this.sprite = 'images/enemy-bug.png';
49     // Enemies are placed randomly in a row (x) in order to space them out
50     this.x = getRandomInt(-101, 505);
51     // Enemies are randomly assigned a row (y)
52     this.y = enemyRows[getRandomInt(0, 3)];
53 };
54
55 Enemy.prototype.update = function(dt) {
```

AWESOME

Great work with the update method.

```
56     // You should multiply any movement by the dt parameter
57     // which will ensure the game runs at the same speed for
58     // all computers.
59     if (this.x >= -colWidth && this.x < ctx.canvas.width + colWidth) {
60         // Uses the speed function to determine enemy speed based off of the c
61         // player level, higher the level, the faster the enemies go
62         this.x += this.speed(player.level) * (dt * 100 - 1);
63     } else {
64         // resets position of enemy to -colWidth so the enemy restarts from th
65         this.x = -colWidth;
```

```
66          this.y = enemyRows[getRandomInt(0, 3)];
67      }
68  };
69
70  Enemy.prototype.render = function() {
71      // Draw the enemy on the screen, required method for game
72      ctx.drawImage(Resources.get(this.sprite), this.x, this.y);
73  };
74
75  Enemy.prototype.speed = function(playerLvl) {
76      // Function used to assign a speed to the enemies based on the player's le
77      if (playerLvl < 5) {
78          // base speed until level 4
79          return enemySpeeds[0];
80      } else if (playerLvl < 9) {
81          // enemies speed up once you reach level 8, moderate speed.
82          return enemySpeeds[1];
83      } else {
84          // for levels above 8, fastest speed is default
85          return enemySpeeds[2];
86      }
87  };
88
89  /*****************/
90  /* PLAYER CLASS */
91  /*****************/
92  var Player = function() {
93      // Initialize player variables:
```

AWESOME

Fantastically done.

```
94      this.sprite = 'images/char-boy.png';
95      this.resetHome(); // sets the player at the 'home' positon for initializat
96      this.level = 1; // sets the player's level to 1 upon initialization
97      this.lives = 10; // player gets 10 lives upon initialization, trust me it
98  };
99
100 Player.prototype.render = function() {
101     // render function proivded, renders the player when called.
102     ctx.drawImage(Resources.get(this.sprite), this.x, this.y);
103 };
104
105 Player.prototype.update = function() {
106     // No update function needed at this time, all movements handled in
107     // handleInput function.
108 };
109
110 Player.prototype.updateLives = function() {
111     // Function to update lives, called in checkCollisions
112     this.lives -= 1;
113 };
114
115 Player.prototype.levelUp = function() {
```

AWESOME

Great work with the leveling system.

```
116     // Every even level you reach, a new enemy is added, increments player's l
117     // as you win a round
118     this.level++;
119     if (this.level % 2 === 0) {
120         allEnemies.push(new Enemy()); // pushes new enemy into allEnemies arra
121     }
122     this.resetHome(); // resets player position to home after you win a round
123 };
124 
125 Player.prototype.resetHome = function() {
126     // Used to set player to home position, called in various functions
127     this.x = colWidth * 2;
128     this.y = yOffsetForChar + rowHeight * 5;
129 };
130 
131 Player.prototype.handleInput = function(key) {
132     // Function recieves key inputs from below, and processes the inputs accor
133     if (key === 'up') {
134         if (this.y === yOffsetForChar + rowHeight) {
135             // checks for a win and resets player if won
136             this.levelUp();
137         } else {
138             //checks if the players next move up is out of bounds
139             this.y -= rowHeight;
140         }
141     } else if (key === 'down') {
142         if ((this.y + rowHeight) <= (yOffsetForChar + rowHeight * 5)) {
143             //checks if the next move down is out of bounds
144             this.y += rowHeight;
145         }
146     } else if (key === 'left') {
147         if (this.x - colWidth >= 0) {
148             //checks if the next move down is out of bounds
149             this.x -= colWidth;
150         }
151     } else if (key === 'right') {
152         if ((this.x + colWidth) <= (colWidth * 4)) {
153             //checks if the next move down is out of bounds
154             this.x += colWidth;
155         }
156     } else {
157         // blank just to cover any unwanted implications
158         // i.e. if other keys are pressed
159     }
160 };
161 
162 /* Check Collisions Function */
163 function checkCollisions() {
164     allEnemies.forEach(function(enemy) {
```

SUGGESTION

Great work with the collision logic but you could have used it as a method on either the player or the enen OOP encapsulation principle.

Example:

```
Player.prototype.checkCollisions = function() {
    allEnemies.forEach(function(enemy) {
       // Checks positions of each enemy on the board,
        // numbers were chosen as an appropriate buffer for the player,
        // if the enemies penetrate the buffer, a collision is recorded
        if (enemy.y >= this.y - 50 &&
            enemy.y <= this.y + 50 &&
            enemy.x >= this.x - 65 &&
            enemy.x <= this.x + 65) {
            // decrement player's lives, return player home upon collision
            this.updateLives();
            this.resetHome();
        }
    });
}
```

```
165         // Checks positions of each enemy on the board,
166         // numbers were chosen as an appropriate buffer for the player,
167         // if the enemies penetrate the buffer, a collision is recorded
168         if (enemy.y >= player.y - 50 &&
169             enemy.y <= player.y + 50 &&
170             enemy.x >= player.x - 65 &&
171             enemy.x <= player.x + 65) {
172             // decrement player's lives, return player home upon collision
173             player.updateLives();
174             player.resetHome();
175         }
176     });
177 }
178
179 /* Initialize Objects */
180 var allEnemies = [
181     // base case has two enemies
182     new Enemy(),
183     new Enemy()
184 ];
185 var player = new Player();
186
187
188
189 // This listens for key presses and sends the keys to your
190 // Player.handleInput() method. You don't need to modify this.
191 document.addEventListener('keyup', function(e) {
192     var allowedKeys = {
193         37: 'left',
194         38: 'up',
195         39: 'right',
196         40: 'down'
197     };
198
199     player.handleInput(allowedKeys[e.keyCode]);
200 });
201
```

- README.md 1
- js/resources.js
- js/engine.js
- index.html
- css/style.css
- LICENSE.txt

RETURN TO PATH

Rate this review

Student FAQ