UDACITY

PROJECT

Classic Arcade Game Clone

A part of the Front-End Web Developer Nanodegree Program

PROJECT REVIEW

CODE REVIEW 6

NOTES

js/app.js 5

```
1
2  //Utility properties for rendering management
3  var RenderingUtility = {
4  bgPatch : 'images/bg-patch.PNG',
5  isPatchActive:true,
6  isGreenGemActive:false,
7  isBlueGemActive:false,
8  isCollisionOopsActive:false,
9  isCollisionOmgActive:false,
10 omgPositionX: 430,
11 oopsPositionX: 10,
12 omgAndOopsPositionY: 10,
13 };
14
15 // Enemies our player must avoid
16 var Enemy = function() {
17     // Variables applied to each of our instances go here,
18     // we've provided one for you to get started
19
20     // The image/sprite for our enemies, this uses
21     // a helper we've provided to easily load images
22     this.sprite = 'images/enemy-bug.png';
23     this.x =0;
24     this.y =0;
25     this.name='bug'
26 };
27 // Update the enemy's position, required method for game
28 // Parameter: dt, a time delta between ticks
29 Enemy.prototype.update = function(dt) {
30     // You should multiply any movement by the dt parameter
31     // which will ensure the game runs at the same speed for
32     // all computers.
33
34     //resets enemy position
35     if(this.x > 505){
36         this.x = 0;
37     }
38
39     //enemy1
40     if(this.name==='bug1'){
41         this.x +=120 * dt;
42     }
43     //enemy2
44     else if(this.name==='bug2'){
45         this.x +=200 * dt;
46     }
47     //enemy3
48     else if(this.name==='bug3'){
49         this.x +=180 * dt;
50     }
51
52     checkCollision(this.x,this.y);
```

REQUIRED

In this project, it is not expected that functions from the global namespace/scope are called to handle a specific object's behaviours/actions. I suggest the following:

```
Enemy.prototype.checkCollision = function () {
  var collisionX = Math.abs(this.x - player.x);
  //Other statements
};
```

then you can call it as `this.checkCollision();` here.

p.s. It is okay if you create a global collision detection function that is not run by passing in `this` in either `Enemy.prototype` or `Player.prototype`

```
53 };
54
55 var checkCollision = function (enemyX,enemyY){
56
57     var collisionX = Math.abs(enemyX - player.x);
58     var collisionY = Math.abs(enemyY - player.y);
59
60     if(collisionX < 65 && collisionY < 70){
61        player.previousPoints = player.pts;
62        if(player.pts < 10){
63             player.pts = 0;
64        }
65        else{
66             player.pts = player.pts - 10;
67             RenderingUtility.isCollisionOopsActive = true;
68             RenderingUtility.isBadSpeechActive = true;
69        }
70        displayPlayerPoints();
71        player.resetPlayer();
72     }
73
74 }
75
76 // Draw the enemy on the screen, required method for game
77 Enemy.prototype.render = function() {
78     ctx.drawImage(Resources.get(this.sprite), this.x, this.y);
79 };
80
```

SUGGESTION

Developer-to-Developer tip

If you want to take your codes to the next level in future projects, why don't you try object inheritance? As you can see, in this project, for example, the `Enemy` and `Player` objects have common attributes/properties (in this case mainly the `x` and `y` properties), so why hardwiring or repeating yourself (remember the DRY - Don't Repeat Yourself - principle) when you can simply use available things?

This is simply a tip to further expand your knowledge of JavaScript and Object Oriented Programming, and it is not required to successfully pass this project.

Here are a couple of resources to start with, if you're interested of course!

http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain
http://www.crockford.com/javascript/inheritance.html

```
81 // Now write your own player class
82 // This class requires an update(), render() and
83 // a handleInput() method.
84 var Player = function() {
85     // Variables applied to each of our instances go here,
86     // we've provided one for you to get started
87
88     // The image/sprite for our enemies, this uses
89     // a helper we've provided to easily load images
90     this.sprite = 'images/char-horn-girl.png';
91     this.x =200;
92     this.y =400;
93     this.pts = 0;
94     this.previousPoints = 0;
95     this.ptsBubblePositive = 'images/speech-bubble-good.gif';
96     this.ptsBubbleNegative = 'images/speech-bubble-negative.gif';
97     this.collisionOmg = "images/collision-omg.PNG";
98     this.collisionOops = "images/collision-oops.PNG";
99 };
100
101 Player.prototype.update = function(){
102 //recenter player if player is past x-axis or bottom of y-axis
103 if( player.x < 0 || player.x >= 500){
```

REQUIRED

Remember to change the `player` variable to the `this` keyword around here too.

```
    player.resetPlayer();
}
//display points if player reaches water zone or top of canvas
else if(hasReachedWaterZone()){
    player.previousPoints = player.pts;
    player.pts+=10;
    displayPlayerPoints();
    player.resetPlayer();
    RenderingUtility.isCollisionOmgActive = true;
    }
}

var hasReachedWaterZone = function() {
    if(player.y<40)
        return true;
    else
        return false;
}

var displayPlayerPoints = function(){
//removes placeholder point of 0 on index.html, and adds updated point
    var pointHeader = document.getElementById("points");

    var ptsPlaceHolder = document.createElement("P");
    var pts = document.createTextNode("Points:"+player.pts+"");
    ptsPlaceHolder.appendChild(pts);

    $("#dummy-pts").remove();
    while (pointHeader.firstChild) {
        pointHeader.removeChild(pointHeader.firstChild);
    }
        pointHeader.appendChild(ptsPlaceHolder);
}

//todo: refactor repeated numbers of Omg, and Oops into the RenderingUtility object
var initCoverCollisionEffects = function (){
if(RenderingUtility.isPatchActive){
    ctx.drawImage(Resources.get(RenderingUtility.bgPatch), omgPositionX, omgAndOopsPositionY);
    ctx.drawImage(Resources.get(RenderingUtility.bgPatch), oopsPositionX, omgAndOopsPositionY);
    RenderingUtility.isPatchActive=false;
    }
}

Player.prototype.displayCollisionEffect = function(){
if(!RenderingUtility.isCollisionOopsActive && RenderingUtility.isCollisionOmgActive){
    ctx.drawImage(Resources.get(this.collisionOmg), omgPositionX, omgAndOopsPositionY);
    ctx.drawImage(Resources.get(RenderingUtility.bgPatch), oopsPositionX, omgAndOopsPositionY);
    RenderingUtility.isCollisionOmgActive = false;
}
else if(!RenderingUtility.isCollisionOmgActive && RenderingUtility.isCollisionOopsActive ){
    ctx.drawImage(Resources.get(this.collisionOops), oopsPositionX, omgAndOopsPositionY);
    ctx.drawImage(Resources.get(RenderingUtility.bgPatch), omgPositionX, omgAndOopsPositionY);
    RenderingUtility.isCollisionOopsActive = false;
    }
}

Player.prototype.resetPlayer = function(){
    player.x = 200;
    player.y = 400;
```

REQUIRED

Here as well!

```
}

Player.prototype.render = function(){
    ctx.drawImage(Resources.get(this.sprite), this.x, this.y);
};

//todo: can use a flag to limit overprocessing bubbles, or ngHide/Show
Player.prototype.renderSpeechBubble = function(points) {
    if(this.previousPoints < points){
        ctx.drawImage(Resources.get(this.ptsBubblePositive), this.x+55, this.y+30);
    }
    else if(this.previousPoints > points){
        ctx.drawImage(Resources.get(this.ptsBubbleNegative), this.x+55, this.y+30);
    }
};

var renderGemReward = function() {
    var posX =240;
    var posY =10;
    if(player.pts >= 100 && player.pts < 250 && !RenderingUtility.isGreenGemActive)
        ctx.drawImage(Resources.get('images/Gem Greene-sm-tp.PNG'), posX , posY);
    else if(player.pts >=250 && player.pts < 500 && !RenderingUtility.isBlueGemActive)
        ctx.drawImage(Resources.get('images/Gem Blue-sm-tp.PNG'), posX, posY);
    else if(player.pts >= 500)
```

```
187         ctx.drawImage(Resources.get('images/Gem Gold-sm-tp.PNG'), posX, posY);
188 };
189
190 Player.prototype.handleInput = function(movePlayer){
191     if(movePlayer === 'left'){
192         this.x += -100;
193     }
194     else if(movePlayer === 'up'){
195         this.y += -90;
196     }
197     else if(movePlayer === 'right'){
198         this.x += +100;
199     }
200     else if(movePlayer === 'down'){
201         this.y += +90;
202     }
203
204     /**Code below allows the player to be rendered via keys as the player
205     //has not reached the water zone. It also fixes the player smear
206     //when player is moved at boundary of canvas on the y-axis.
207     //update function above resolves rendering if the player has reached the water zone.
208     //The smear effect of player at the boundary of the y-axis influenced this coding
209     note: code below can be refactored/consolidated w/ update function above
210     **/
211     if(!hasReachedWaterZone()){
212     //check if players bottom of canvas
213         if (this.y > 400){
214             player.resetPlayer();
215             player.render();
216         }
217         //otherwise render player
218         else
219         {
220             player.render();
221         }
222     }
223 }
224 // Now instantiate your objects.
225 // Place all enemy objects in an array called allEnemies
226 // Place the player object in a variable called player
227 //todo: can  somehow refactor these objects into a one global init function, maybe with IIFE
228 var initEnemyAndPlayer
229 var enemy1 = new Enemy();
230 enemy1.x = 50;
231 enemy1.y = 55;
232 enemy1.name = 'bug1';
233
```

SUGGESTION

If you are interested in Game Development, perhaps the below references can help you to code games like a pro! :-)

References

http://codeincomplete.com/posts/2013/12/10/javascript_game_foundations_state_management/

http://stackoverflow.com/questions/18038502/how-to-code-a-html5-game-with-distinct-game-states

http://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867

https://github.com/jakesgordon/javascript-state-machine

```
234 var enemy2 = new Enemy();
235 enemy2.y = 145;
236 enemy2.name = 'bug2';
237 var enemy3 = new Enemy();
238 enemy3.x= 320;
239 enemy3.y= 225;
240 enemy3.name = 'bug3';
241
242 var allEnemies = [enemy1,enemy2,enemy3];
243
244 var player = new Player();
245 // This listens for key presses and sends the keys to your
246 // Player.handleInput() method. You don't need to modify this.
247 document.addEventListener('keyup', function(e) {
248     var allowedKeys = {
249         37: 'left',
250         38: 'up',
251         39: 'right',
252         40: 'down'
253     };
254     player.handleInput(allowedKeys[e.keyCode]);
255 });
```

README.md 1