Adv JS Ca1 Report

Space Invaders

Matthew Oppermann - N00147265

Introduction

The JavaScript game that i designed is a space Invaders style game that has the player controlling a ship, firing at asteroids and meteors as they make there way towards you. The player has 3 lives to begin with, and must destroy all the asteroids while also dodging oncoming bullets from an enemy ship using the arrow keys. If the players ship is hit by a bullet, a life is deducted. If the player lives reach 0 then it is game over. Similarly if any of the asteroids reach the end of the field then it is game over. The game features a special object which is a fast paced meteor that bounces around the games screen. If this meteor is shot down, the player is rewarded with extra points. Whether ending in success or failure, the player has the option of submitting his score along with a username to the database.

Functionality

Asteroids

The asteroids that make up the main enemy are stored in an array. This array was set up with a for loop so that every time the loop ran, a new asteroid was created from the Asteroid class. This asteroid was stored in the array as a new element. This allowed me to set the create the amount of asteroids in the game desired without having to set and call the Asteroid over and over again.

```
for (let i = 0; i < 11; i++) {
    let x = random(width);
    asteroids[i] = new Asteroid(i * 40, 35, 30);
}
```

In order for the Asteroids to pose a threat, they had to be able to move. I did this by setting a variable xdir which could either be assigned a negative or positive number depending on whether the asteroid was meant to move left or right. Below we can see in the shift down function, the xdir is multiplied by minus 1. This is to get asteroids moving back left when they hit the edge on the right. The y coordinates are also lowered down by the size of the asteroids radius.

```
shiftDown() {
    this.xdir *= -1;
    this.y += this.r;
}
move() {
    this.x = this.x + this.xdir;
    this.y = this.y + this.ydir;
}
```

Below is the code that checks through the asteroids array to see if they have touched the edge. This is done by checking if their x value is either greater than the width(for right edge) or less than 0 (for left edge). If they hit it then edge is set to true, and depending on what side it is, the xdir is set to a - or a + number.

```
let edge = false;
//loops through asteroids array and calls funcrtions for each asteroid.
for (let a of asteroids) {
    a.show();
    a.move();
    //checks if the asteroids hit both sides of the canvas
    if (a.x > width || a.x < 0) {
        edge = true;
    }
}
```

Bullets

Similar to the asteroids, the ships bullets were stored in an array. This allowed the reuse of the bullet class without having to rewrite it.

Below is the function hits, that checks if the bullet has hit an asteroid. This is done by getting by checking if the bullets x, y and radius is greater than then the asteroids x, y and radius values. If the bullet hits function is true, then the asteroid is spliced from the array and removed from the game. The same is done to the bullet, its spliced and removed so that the ship can fire again.

```
hits(asteroid) {
    let d = dist(this.x, this.y, asteroid.x, asteroid.y)
    if (d < this.r + asteroid.r) {
        return true;
    } else {
        return false;
    }
}
```

Meteor

A meteor object was introduced as a special target that granted the user extra points. The meteor is a fast bouncy object that hit off all corners of the screen. This bounce was done by function which checks if the ball is hitting the edges. If it does then an appropriate speed is applied to either the x or y values sending the meteor in the opposite direction.

```
bounce() {
    //setting the bondaries so that if the meteor hits an edge, the direction is changed
    if (this.x > width || this.x < 0) {
        this.xspeed = this.xspeed * -1;
    }
    if (this.y > height || this.y < 0) {
        this.yspeed = this.yspeed * -1;
    }
}
```

Boss

The boss is a rival shift that fires down upon the user. The function below in the boss class moves the boss ship to follow the players ship. This is done by again getting the x coordinates plus the width of the boss and checking to see if it is less than the players x coordinates plus the width. If it is then 3 is added to the ships x coordinates so to move the ship and eliminate the gap.

```
move() {
    //cheking if the distance between the ship and the boss is greater tha
    if (this.x + this.w / 2 < ship.x + ship.w / 2) {
        //if it is then it is moved by 3
        this.x += 3;
    }
    //this is for the opposite direction
    if (this.x + this.w / 2 > ship.x + ship.w / 2) {
        this.x -= 3;
    }
```

Lives (bug)

There were bugs in the player and boss lives that were unresolved. This was with the colision between the players bullet and the boss' lives, where if hit, the boss would lose a life. Instead the boss lost a number of lives instead of just one.

Score - Database

Once finished playing the player can submit their username and score to the high score database. This was done by first having an input box, whose value was taken and inserted along with the score into the database.

```
const button = document.getElementById('button');
console.log(button);

button.addEventListener('click', function(e) {
    let name = document.getElementById('name').value;
    fetch('/score', {
            method: 'PUT',
            body: JSON.stringify({
                name: name,
                score: score
            }),
```

Below is a confirmation message in node and mongoDB that the data was received.

```
Data received: {"name":"Sean","score":33}
Data received: {"name":"Rich","score":21}
Data received: {"name":"Julz","score":53}
```

```
_id: ObjectId("5a68dbcfedfe6b0bacc65f1b")
name: "Sean"
score: 33
```

```
_id: ObjectId("5a68de54edfe6b0bacc65f1c")
name: "Rich"
score: 21
```

```
_id: ObjectId("5a68de80edfe6b0bacc65f1d")
name: "Julz"
score: 53
```

Conclusion

Overall I was happy with how the assignment turned out.I was pleased with the functionality of the ship its bullets and how the asteroids moved and lowered along the screen and eventually how they were destroyed. The boss was another feature that I liked, how it followed the user from location to location. The collision detection between player and boss was an irritating part as other collision detection used in the project worked perfectly.

The high score was another plus side, with the score along with the username entered by the player being successfully saved to the database.