How to Build Your First Game using ReactJS

First go to paransonthalia.com/tutorials.html and download the ReactJS tutorial

1. Let's start with going over some of the structure of ReactJS
    1. React only has one JavaScript file in which you also write your HTML contrary to a normal website which has an html and JavaScript file
    2. In your JavaScript file you will have your main class in which you have any necessary functions/variables and you have your most important render method
    3. Inside of the render method, you will return a single html element (generally a div) in which you have anything else that should be main on the page
        1. This includes objects like headings, navbars, and more
    4. Then for most of the heavy lifting, you will declare multiple classes. These will include different elements which get changed often. For example on Uber's app, the map is a new class, then everything on the map is a subclass under that new class.

1. So we are going to be making a space shooter game. I have created some template code and we will be filling in the blanks together. If there is time left at the end, we can add more features to the game!
2. First open up the starter project in the folder that you have downloaded
    1. Open up the starter project file in Sublime Text
        1. The file that we will be editing is under ./src/App.js
3. Then open up the command line by opening up the starter project in windows explorer. Now push the Shift key and then right click. This will open up a menu and you will click on "open in command line"
4. Now type in the following command
    1. `npm start`

5. Let's start simple with adding a simple background image.
    1. For this step we will not be using App.js and we will instead be using ./public/ index.html
        1. In this file you will see the following on line 24
            1. `<body> <!--Begin Step 5—>`
        2. Change this line to read
            1. `<body background="./background.jpeg"> <!--Begin Step 5—>`
    2. What we have done here is that we are using pure HTML to set the background image of our game. We can change this image to anything and you can go

© Paran Sonthalia 2018

ahead and download any image that you want and set that to the background of your game!

6. Next, we will be adding enemies to our game.
    1. Locate the following line
        1. // Begin Problem 6
    2. We could add more enemies by copying and pasting the same line over and over again, but the more efficient way to do it would be to use a loop. We can use a for loop for this task. Start by typing the following line.
        1. **for** (**let** i = **0**; i < **10**; i++) {
    3. Here we are creating a loop which will run 10 times. Inside of this we want to create a new Enemy. How we create this Enemy is the heart and soul of ReactJS. We can just write our own HTML code which will then use the class that is defined below.
        1. enemies.push(<Enemy x={Math.floor(Math.random() * (window.innerWidth - **150**))} y = {Math.floor(Math.random() * **200** + window.innerHeight - **400**)} image={"./enemy.png"} getY={**this**.getY} getX={**this**.getX}/>)
    4. A lot of this is overcomplicated and you don't need to understand it right now, but here are the important parts. Where we do "<Enemy" we are adding a new Enemy to our game. We are giving the Enemy new x and y values through the code that does "x=" and "y=". The rest of this is just extra information that makes this all work.
    5. Lastly, don't forget to close the for loop with the following line.
        1. }

7. Now we will be making the enemies move from left to right randomly
    1. Locate the following line
        1. // Begin Problem 7
    2. What we need to do here is change the x value of our enemy based on a random direction to move.
    3. We already have a direction declared above which randomly sets a variable to true or false based on which way to move.
    4. What we will use is called an if statement and it will run a certain block of code if one condition is satisfied and it will run the else case if that condition is not satisfied.
    5. So, let's start with the following code
        1. **if** (**this**.state.direction === **true**) {
            this.state.x += **5**

```
} else {
    this.state.x -= 5
}
```

6. What this does is that it checks if the direction variable is true and if it is then it increases the x position of the enemy by 5 and if the variable is false, then it decreases the x position by 5
7. Let's go to the browser and run this code.
8. Notice how the enemy keeps going off to one side and does not bounce off and go in the other direction.
9. To fix this, we will need to update our direction variable if our x value is too big for the right side and too small for the left side
10. Go to the next line and type the following

    1.
       ```
       if (this.state.x < 20 || this.state.x > (window.innerWidth - 90)) {
           this.state.direction = !this.state.direction
       }
       ```
11. We are checking if the x is less than 20 or (||) if the x is greater than the width of the screen. If it is, then we change the direction to the opposite of what it is. This way the enemy will start moving the other way.

8. Now we have our enemies moving and our ship can shoot projectiles! All we need is the ability for our ship to move.
    1. Start by locating the following line
        1. `// Begin Problem 8`
    2. We already have variables that will turn to true while a certain arrow key is pressed and false when it is depressed. So what we need to do is move our player in the respective direction while the certain key is pressed.
    3. Start with the following line and try to figure out the remaining lines.

        1.
           ```
           if (this.state.movingUp) {
               this.state.y += 1
           }
           ```
    4. We are moving the y position of the ship by one if the up arrow is pressed. Based on this we can also move the ship in the rest of the directions with the following code.

        1.
           ```
           if (this.state.movingLeft) {
               this.state.x -= 1
           }
           if (this.state.movingRight) {
               this.state.x += 1
           }
           ```

© Paran Sonthalia 2018

```
if (this.state.movingDown) {
    this.state.y -= 1
}
```

9. Now we have a functioning space shooter game!

Challenges

If you finish early, or want to continue this at home, try out these challenges.

1. Make the enemies move down as the game continues
2. Make a “You Win” screen when all of the enemies disappear
3. Make a “You Lose” screen if the ship comes in contact with any enemy

© Paran Sonthalia 2018