Lab 3

In this lab we will build a mini browser-game, and also get a bit of practice creating classes. First, we'll need to update your project.

Updating your project

- Start up GitKraken.
- Make sure that you have committed all files related to your project. When you look at your project in GitKraken, you should be seeing no WIP section and no modified files.
- Right-click at the "upstream" item in the "REMOTE" section on the left side. Choose "Fetch upstream". You should now be seeing that your graph in the middle has two "master branches". One corresponds to "my repository" (the upstream) and the other corresponds to your master branch with the changes you've made on it for Labs 1 and 2. This should be the "active" branch, with a little checkbox next to it. These two branches should appear to "deviate" from a common start.
- Right-click *my* master branch (upstream). You can do so either in the main window or in the upstream->master section in REMOTE on the left. Choose "Merge upstream/master onto master". You should see the two branches merge into a new "merge" commit.
- You are now ready to work on Lab 3! After you add your changes, save and create a commit in GitKraken, then push your changes.
- You will need to repeat these steps when a new lab is released.

Assignment

In your project you will find a Lab3 folder and a number of files:

- `index.html` file within that folder. This will be used to test your progress, and we will describe that process in a moment.
- `controller.js` is code provided by me that drives the testing and playing of the resulting game. You should not need to do anything with it.
- `game.js` is a "module" that holds the `Game` class. You will need to implement the details of this class.
- `circle.js` is a small "module" that holds a `Circle` class representing circles. You will need to implement a constructor for the class.

Running a local server

In order to test the application, you will need to open the `index.html` file *as if it was a file served from a server*. In order to achieve that, you will need to be running a local server.

If you are on the lab computers, then open up a terminal window and navigate to the Lab3 folder of your project. In that folder, execute "http−server" from the terminal. You should see a response which will look something like this:

```
Starting up http−server, serving ./
Available on:
  http://127.0.0.1:8080
  http://10.83.0.52:8080
Hit CTRL−C to stop the server
```

Keeping that terminal window open, open up your web-browser and in the location address write:

```
http://127.0.0.1:8080/index.html
```

where the numbers should be whatever your system is reporting.

If you want to do this on your own machines, you need to either know a way to run a local server, or you need to install the http−server package from NPM. The instructions to do that would likely be:

```
npm install −g http−server
```

You may need to install Node and NPM if you don't have those set up yet, and you may also need to run the above command as administrator.

You can stop the server at any time by hitting Ctrl-C on the terminal.

Assignment questions

The questions you need to answer follow. When you are satisfied with your answers, save all your files and make a commit and push it.

When you open up the index.html file, it should show you a red banner with what still needs to be done to make progress. Briefly the steps are as follows:

1. In the game.js module, you need to implement the details of a Game class. The goal of the Game class is to keep a score as the game is played. Your constructor should be accepting one argument, which is the "maximum number of points". You will need to choose a good variable name for it. Your constructor should then store this value in an instance variable.
2. Your Game class constructor will also need to create an instance variable called points and initialize it to 0.
3. Your Game class needs to have a method isOver(), which returns true if the number of points has reached or exceeded the maximum number of points.
4. Your Game class needs to have a method reset() which resets the number of points to 0.
5. Your Game class needs to have a method hit() which increments the number of points by 1.
6. Your Circle class needs to have a constructor that takes a single color parameter. It then stores that color in an instance variable named color.

7. Your Circle class constructor needs to also set values for variables x, y and radius, which are not provided as parameters. Your x should be a randomly generated number between 0 and 600, your y should be a randomly generated number between 0 and 300, and your radius should be a randomly generated number between 10 and 30. The Math.random() method can help you, as it will generate a random number between 0 and 1. You will need to transform that number suitably fit the desired scales.

Once you complete these steps, you will be able to play the game. Every 2 seconds a new circle will appear on the board, and you have 2.5 seconds to click on it before it disappears. If you do, then you get a point. You need to reach 20 points to win. Feel free to adjust those numbers in the controller.js file if you would like a bit more challenge.