NANDA KISHORE nkk263@nyu.edu +1 (718) 679 7382

RAPHAEL SNAKE

Working app is running at https://nandukalidindi.github.io/raphael-snake on my almost portfolio website(https://nandukalidindi.github.io)
Codebase: https://github.com/nandukalidindi/raphael-snake

Minimal snake game using **Javascript** and **RaphaelJS**(http://dmitrybaranovskiy.github.io/raphael/)

RaphaelJS is a **SVG** renderer and almost similar to **Canvas** but gives the flexibility to control individual elements for a better control of events and animations.

However the render strategy would still be the same for this application irrespective of what is used
1. Render food, snake body and board
2. Clearing the paper object on each interval and re-drawing with the new respective positions.
3. Removing the tail and adding the new possible position as head to the snake

TOOLS:

- HTML, CSS and RaphaelJS are used to draw objects on the screen
- Javascript(ES6) for functionality
- Webpack with babel loader is used to build the application and to use ES6 features to make it cross browser compatible.
- JSDoc for documentation

IMPLEMENTED FEATURES

- Eat food and increase in size
- Stopping the game on crossing the boundaries
- Stopping the game when snake intersects with itself
- Score update
- Play / pause the game at any instant
- Reset / Restart game
- Animations on the food object and score
- Option to change the colour of food and board

TO BE IMPLEMENTED

- Undo feature (Reverse the gameplay to a non-danger state)
- Provide options to change the snake color, snake shape etc. and option to change **frame rate.**
- Option to play on a rather free board without boundaries, where in crossing a boundary will let the snake appear from the opposite boundary.
- Polyfill for Safari because it does not seem to support the javascript **animate API**.

NANDA KISHORE nkk263@nyu.edu +1 (718) 679 7382

CONVENTIONS

Variable and method names are deliberately named a little relaxed and out of the usual manner to make the code more relatable to Snake game. Please ignore if they are way too weird.

Tried to put in as much ES6 sugar as possible.

There will not be a single **IF ELSE** statement in the submitted code, just wanted to try out a coding style that I have been wanting to experiment with recently. Since I can't try it out on production code bases, I tried it out here. Please ignore if it is not preferred.

All entities are packed in ES6 classes. The constructors are a little different from the usual standard. Instead of writing multiple **this.property = property**, a mass assignment approach is chosen.

All methods are documented with as much explanation as possible.

INSTRUCTIONS:

- Press **spacebar** or click on the **play** icon at the top.
- Everything else is similar to how the Snake game is.
- To pause game play hit **spacebar** or click on the **pause** icon, click again to resume.

BUILD INSTRUCTIONS:

If the app needs to be modified then **npm install** on the project root and **npm start** to start webpack watch so that new bundle.js is built on every change an open index.html on a browser page

- npm install
- npm start
- Open index.html on browser

POSSIBLE BUGS:

Color pickers do not work on Safari but that does not tamper with game play

Showing **GAME OVER** when the respective scenarios occur

HAPPY PLAYING!

Development of the app is done on Chrome Version 61.0.3163.100 (Official Build) (64-bit)

Please let me know anything is ambiguous.

PS: I don't personally have any inclination to this coding style or any, I love to take up anything that looks great and understandable.