EECS 493 2019 - Assignment 2 - Blaster Game

195 points (with 12 bonus points possible)

Due: November 3, 2019 (11/3) @ 11:59pm

Description

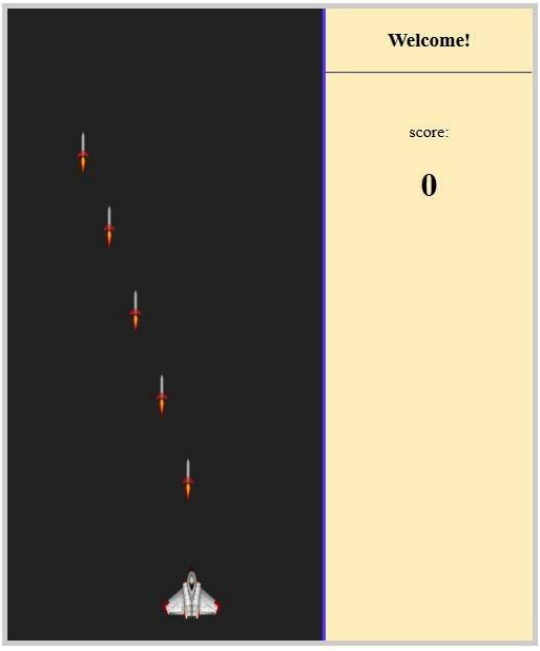
In lecture, we went through how to develop a simple asteroid blaster game using HTML, CSS, and Javascript/jQuery. In this assignment, you will be given starter code of the same game on Canvas (Files → Assignments → Assignment2 → A2_StarterCode.zip). Your task is to modify the existing code to add functionality to fulfill the following requirements so that the game will be ready for deployment.

START EARLY, SINCE UNDERSTANDING THE CODE MAY BE CHALLENGING!

Starter Code

The starter code folder contains the following folders and files:

- **img/ folder.**
  Contains images for index.html.
- **scripts/ folder.**
  Contains .js files for index.html.
- **audio/ folder.**
  Contains audio files for index.html.
- **style/ folder.**
  Contains CSS files for index.html.
- **index.html.**
  The main html page.
- **README.txt**
  Contains explanation of the starter code.



As the screenshot on the right shows, the starter code contains the following functionality:

- 2D ship movement
- "Fireable rockets" that can destroy asteroids they collide with
- Asteroids that can be "spawned" based on user input, and fall vertically
- Score tracking
  - Single-session scoring (clears on refresh)
  - Score per asteroid is inversely proportional to size (smaller==harder==more points)

You can control the game using the following keys:

- Arrow keys: move ship
- Spacebar: fire rocket
- Shift: spawn an additional asteroid

Requirements (Total 195 pts + Bonus 12 pts)

In this assignment, you are asked to implement ten basic and two advanced functionalities from the starter code. You will need to list functionalities you implemented within a text file (details see Submission section below), so we do not miss any functions you created. There are three levels to the game. When the user hits 10,000 points, they advance to the next game level.

General (115 points)

(5pts - Accuracy) Count and display the accuracy rate in percentage (e.g. 22%, **no floating point**) the number of asteroids destroyed / the number of rockets launched.

(15pts - Setting Panel) Create a button labeled “Open Setting Panel” on the scoreboard window (right). If a user presses the button it will show the “setting panel” that contains game-setting parameters that are specified in the following subtasks. If a user presses the button again, the setting menus will hide. The setting panel should have a button called “Update” on the bottom. Once opened the label of the button should change to “Close Setting Panel” and so on. Any update on parameters should be effective when the user presses the Update button (not when they enter it). Once the Update button is pressed, the panel should hide again. Note that any change in parameters are local (javascript variables), meaning if you refresh the page all the parameters will have default values.

(20pts - User Experience, Style, Misc) Note that we are not going to read your code. Rather, we are going to play the game as a user. We will evaluate the general layout, the overall user experience of the game, and how decent it looks and feels.

(15 pts) Please include a “trap door” for easier debugging and grading. That is, if you press the 'L' key, your game will automatically advance to the next level. This way, you don't have to keep playing the game to get to 10k points before advancing, so this is really helpful for debugging (and turns out, helpful for grading too! :D).

(10pts - Game Over Panel) When the game ends (GAME OVER), display a game-over panel on the left side:

- the score of the user on top of the game play window.
- A button labeled “Go Back” that allows the player to go back to the splash screen.
- The button should initialize game status.
    - The ship and its position
    - The score
    - The accuracy

(10pts - Sound Effects) Add sound effects for:

- rocket launching
- the ship - asteroid collision sound
- game over when the game is over,
- intro music whenever the splash screen appears.
- Transitioning to a new level

You can find some useful sound effects files in https://www.freesound.org. Feel free to use basic sound files we post. Make sure you load audio file only once, to save on memory. Don’t worry if it does not sound if you launch rockets too quickly. Make an option to mute all sound using a checkbox in the setting panel. By default the checkbox should be checked, meaning it is muted initially.

(20pts - Splash Screen) At the beginning of the game, create a splash screen on the left side. Display the game instructions (key mapping) and the START button on top of the gameplay screen (overlaid). When the player clicks “START”, the game will start. When the game is not running, a user can move the ship but cannot launch rockets and there should be no asteroids spawning. Note that there should be life icons only when the game is running. Refactor your code so far accordingly. (Hint: Add a “state” variable that indicates the state of the game. e.g. running, game over, initial.)

(20pts - Visual Effects) Add visual effects for ship-to-asteroid collision

- For example, if an asteroid hits the ship, there will be an explosion mark on top of the ship. The visual effect should disappear in a second.
- (Hint) Note that you should not worry about the moving position of the explosion visual.

Level 1 -- Automatically Spawn Asteroids (40 points)

(30pts - Spawning Asteroids Automatically) Currently, pressing the SHIFT key will spawn (create) an asteroid. *Your task is to make spawning asteroids automatic.* Create an input box on the setting panel that takes a number ***S***, the average number of asteroids spawned per second. The default number should be 1. Alert the user if they enter anything other than a number in the range [0.2, 4]. Once a user updates the number, spawn an asteroid with the interval determined by *S*. Add a randomness value of +/- 50% of the interval. For example, if the interval is 1 second, the interval should be a random number between 0.5 and 1.5 seconds. Note that setInterval will use a fixed number (even though the fixed number can be random) across the executions. Print out the random interval used in the console using the console.log. Use the "placeholder=" property to guide the range. If nothing is entered, it should not update the current value.

(10 pts -- Asteroids spiral) As the asteroids fall down, they will spiral (spin) . You can decide the spin rate.

Level 2 -- Shield as Defense, Random Speeds (40 points)

(20pt - Shield) Per every ***M*** asteroids destroyed, create a shield item that moves (and behaves) like an asteroid. However, if the ship collides with the item, it will create a shield on the ship. The shield will protect the ship from one collision with an asteroid. e.g. if the ship with the shield collides with an asteroid, the ship is not destroyed but the shield disappears. Use 10 for default value of M (itemRate).

b. Note that a shield can be destroyed (as it falls) by a rocket
c. (Hint) Note that you should not worry about the moving position of the shield.

(20 pts - Random speeds) Since this level is a bit harder, make it so that 1 out of every 3 asteroids spawned suddenly speed up by a factor of 5 (So they go 5x the speed). If you successfully shoot these faster asteroids, your point value increases by twice the amount it would have.

Level 3 -- Asteroids Attack You! (20 points)

(15 pts -- Asteroids aim) Since this is the last level, let's kick things up a notch: in

this level, randomly, an asteroid will increase speed by a factor of 5x and will now *aim towards the ship*. Instead of falling straight down, the asteroid will instead attack the ship, wherever it is.

(5 pts -- Dual Vision) In fact, not only will some asteroids randomly aim for the ship, but now you get to have TWO ships that appear side-by-side and shoot rockets. This means twice the firepower. Both spaceships will be adjacent to each other and will respond in the same way to key movements. E.g., if left arrow is pressed, both spaceships will move left.

Bonus Tasks (12 points)

(BONUS - 3pts) This is a bonus task. Current movement is discrete per a keydown event. Make the movement continuous. For example, if a user holds the right arrow key, the spaceship will move "constantly" at the speed of 50px per second (1px per 20 ms).

a. Hint: use setInterval.
b. This may be more difficult than you think so you may want to focus on other assignment aspects before working on this.

(BONUS - 9pts) Be creative! Add any functionality that you want. Depending on the "cool factor" (and complexity), we will evaluate the bonus feature *subjectively* for a total of up to 9 points. Simple features will only get one or two points.

Important Notes

1. External libraries, other than jQuery and Bootstrap, are not allowed.
2. We care most about your game's functionality, but we still want you to write clean code. Try to avoid mixing JavaScript and JQuery. Choose which convention you prefer and stick with that for the project.
3. We will be testing your project in the most current version of Chrome, so you may use any ES6 conventions you like.
4. Avoid using any image or audio file that is not open to the public. If used, specify them in the README file and include the content with your submission.
5. Use CANVAS Discussion and Discussion Sections for questions.
6. **START EARLY. THIS ASSIGNMENT IS DIFFICULT!**

Submission

Submit one .ZIP file that contains one folder and one text file to Canvas.

Inside the folder, you should need to include:

1. Javascript file with the functionalities implemented.
2. All other files that were created or modified.

Name the text file README_[your_uniqname].txt file. In this file, please list the following:

1. The spec functionalities you were able to implement.
2. Any bonus functionality you implemented. Explain how you did so.
3. Credit/References to any resources (image, music, etc.) you used in the project but not created by yourself

Please make sure that you successfully implement all the functionalities you listed in the file.