CSC543 Fundamentals of Internet Programming

Fall 2016

Assignment 02

Due Date: Monday 11.07.2016 11:59:59pm

Wall2Wall: A JavaScript Game of Bouncing Balls

Introduction:

In this assignment, the students will build a JavaScript game from provided skeleton code (don’t use any third-party JavaScript libraries). Writing this game tests the knowledge of the students in various topics of the scripting language; from variable scopes, arrays and objects, functions, to graphics, animation, and DOM manipulation.

Description: The user begins with adding balls and collision targets (along the edge of the play area). The game has to support up-to five players that should be controlled by a variable in the script. The first created ball is owned by player1, the second one by player2, and so on, with the ball ownership changing cyclically (1, 2, 3, 4, 5, 1, 2, …). The targets are not owned by any players. A player can own multiple balls, with his/her score being a ratio of the total score and the total number of balls owned by the user. Targets and balls can be added in the middle of the game. Each of the balls and the targets initially start with some point attachment. The webpage should also show individual information areas for each of the players showing their corresponding scores and ball counts.

After the game starts (the user clicks a button to start it), the balls start moving with random velocities and when collided with a target, awards the corresponding player some points. Upon collision, both the target and the ball lose some of their points. The game runs for a predefined time (should be easily modifiable from the script), and after time is up, animation is frozen and an alert dialog should pop up showing which player won (has the highest points).

Requirements:

**The students should start from the skeleton code provided (assignment02\_starter.html)**

The completed assignment should have the following features (in addition to the ones already in the skeleton code):

1. Each of the targets should initially have a random point associated with it. For vertical targets, it should be between 100 and 200 (inclusive for both ends), and for horizontal targets it should be between 50 and 100 (inclusive for both ends).
2. When the targets are drawn, the points currently (remember, they can change over time as a result of collisions) associated with them should be displayed near each target (see Fig 2. For the exact graphical requirements).
3. The first ball is assigned to player1, the second one to player2, …, the fifth one to player5, the sixth one to player1 again, and so on in cyclic ordering. The balls associated with each player will contribute towards his/her score.
4. The balls should display the random points (between 0 and 10, inclusive) associated with them.
5. Balls and targets should be allowed to be added anytime during the game.
6. The user should be prevented from creating targets that overlap with each other for a given orientation.
7. Each of the players should have two information panels showing: (i) number of balls associated with the player, and (ii) the score.
8. Once the game starts, the button for starting the game should be disabled.
9. During the gameplay, each time a ball hits a target the following things should happen:
   1. The player should be awarded points equaling to: (points for the target) x (points for the ball). Current values should be used in the calculation, not initial ones.
   2. The points of the target should be reduced by an amount equaling to: 0.5 x ( the current points associated with the collided ball)
   3. The points of the ball should be increased by a factor of 1.1
   4. The current score of the user should be updated to reflect the newly awarded points. The score of a player is equal to: (points scored by the player) / (number of balls owned by the player at that point of the game). Show at most two digits after the decimal point.
   5. The information area showing the target’s current points should flash momentarily to emphasize the impact ( hint: assign a different background color to the information area for about 200 milliseconds before restoring the original)
10. The game should always update the information area for the players with their updated scores and number of balls.
11. After a predefined time, the game should stop moving the balls and display which player has the highest score using an alert window.

Here are some screenshots from the game showing its different states:

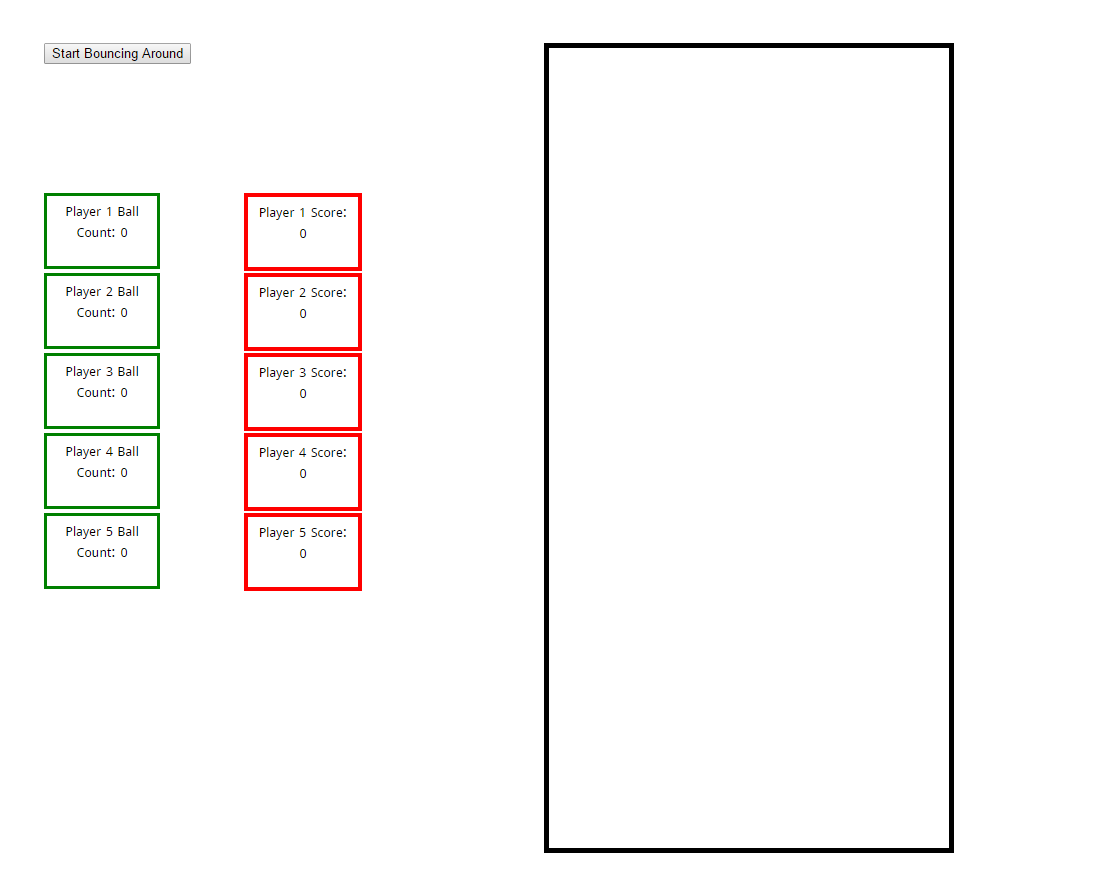


Fig 1: Initial state of the game

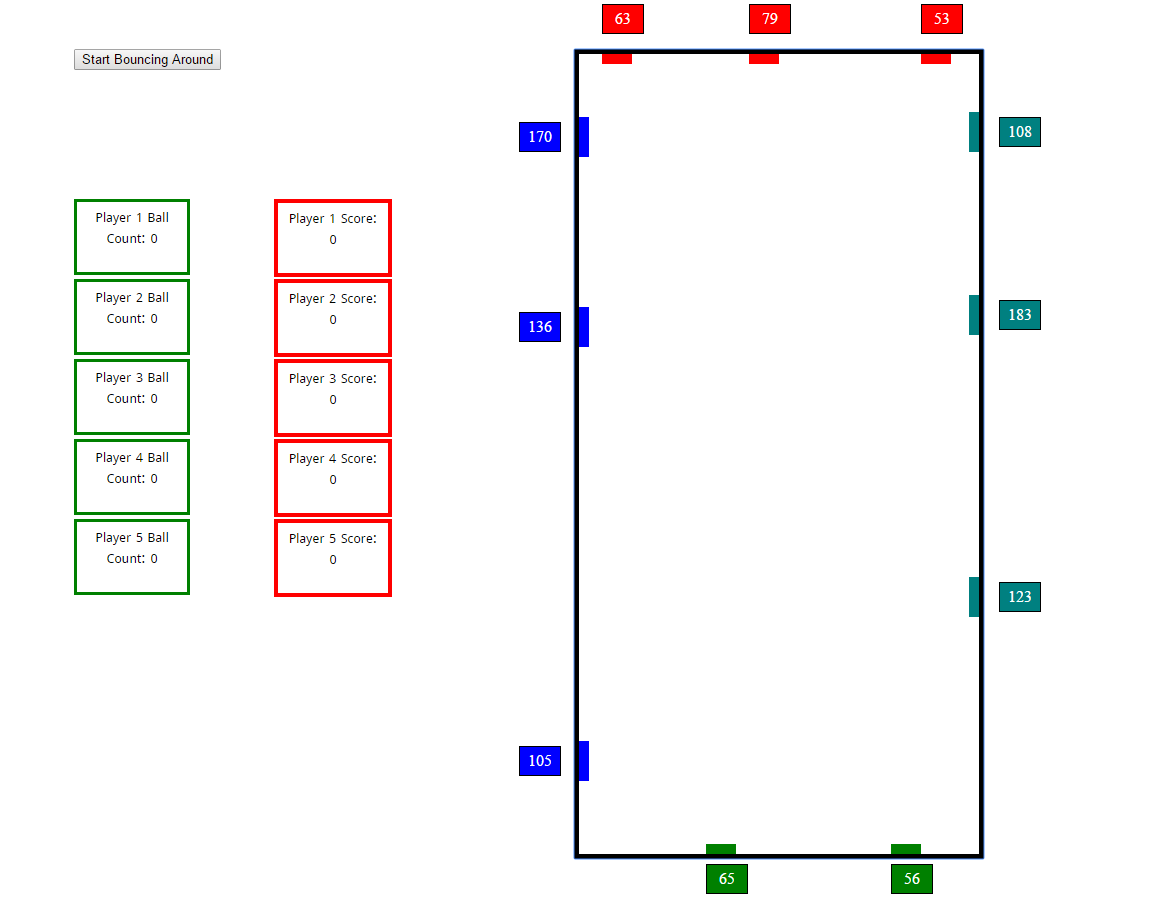


Fig 2: After targets have been added (yet to add balls).

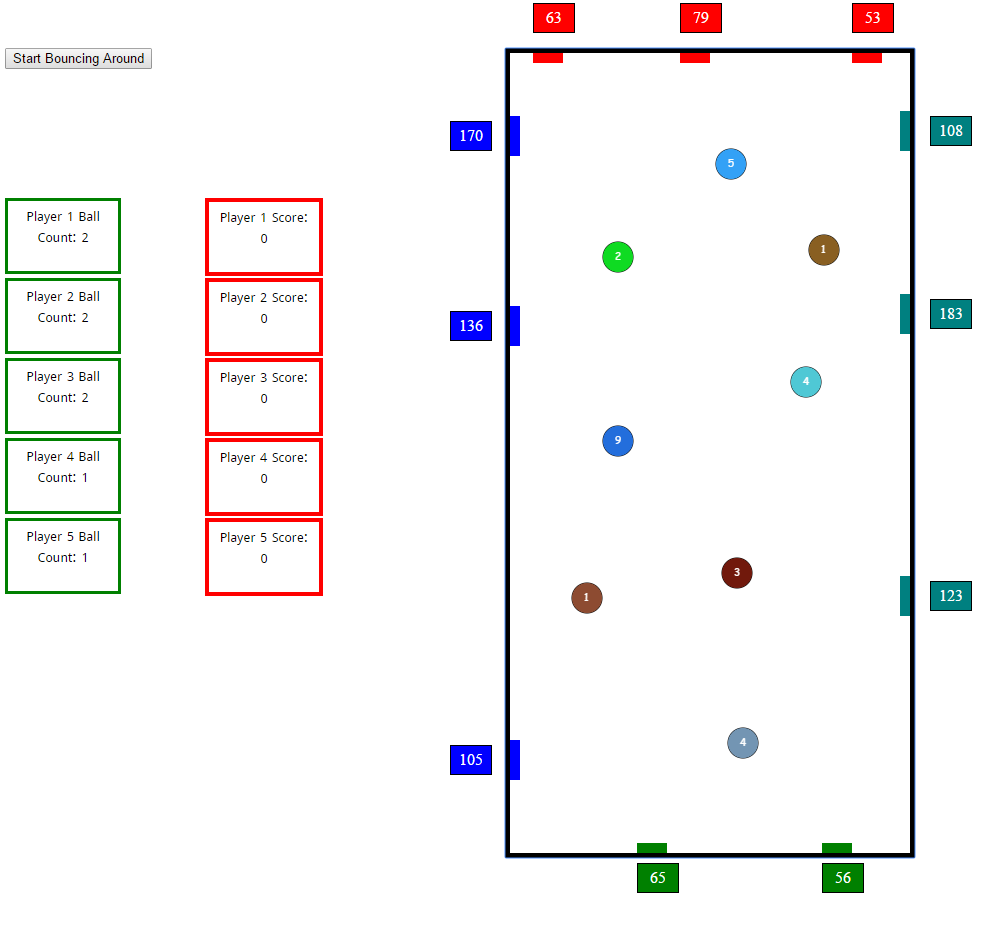


Fig 3: Balls added. Notice how player 1, 2, and 3 have added two balls each, and player 4 and 5 have added only one each. Game has not been started yet.

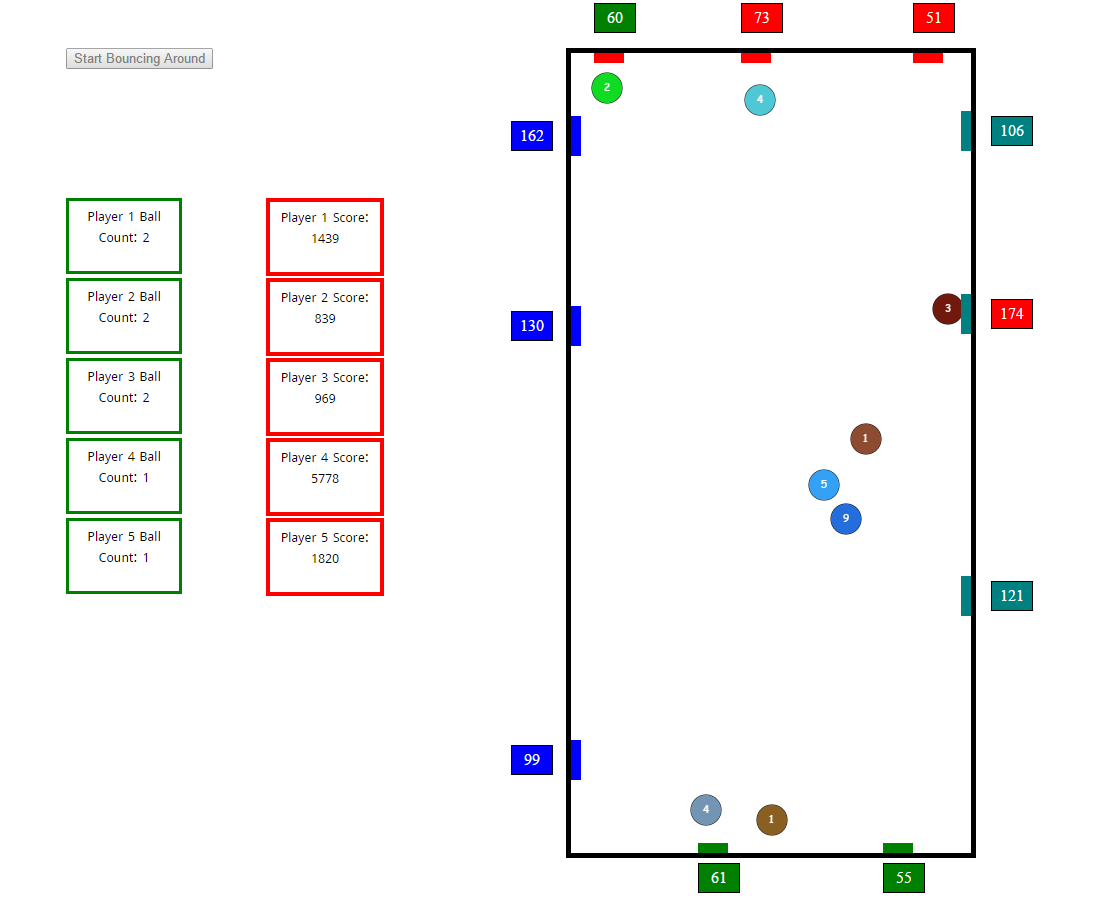


Fig 4: A game going on. Notice that the red boxes are showing the scores of the players. Also notice that the button has been disabled. Moreover, we can also see that some of the targets (the middle one on the right edge, and the leftmost one on the top edge) are flashing colors different than what they have normally.

**Submission Instructions:**

Assignment submission should include:

1. The html file containing the game, and
2. .js and .css files ( if JavaScript code and/or CSS have been stored separately)
3. This (these) files should be compressed together using the zip utility and should be named as: <student\_last\_name>\_CSC543-01\_F2016\_Assignment02.zip
4. The zip file should be emailed to [islamm2@southernct.edu](mailto:islamm2@southernct.edu) by the deadline ( 10% penalty each day after the deadline up-to 5 days late at most; after that no submission is allowed )

**Useful resources on JavaScript:**

Mozilla Developer Network:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference>

W3Schools:

<http://www.w3schools.com/jsref/default.asp>

Sitepoint:

<http://reference.sitepoint.com/javascript/domcore>

--------------------------------------------------- x -------------------------------------------------------