GUI Assignment

Implementing a Checkerboard-Output

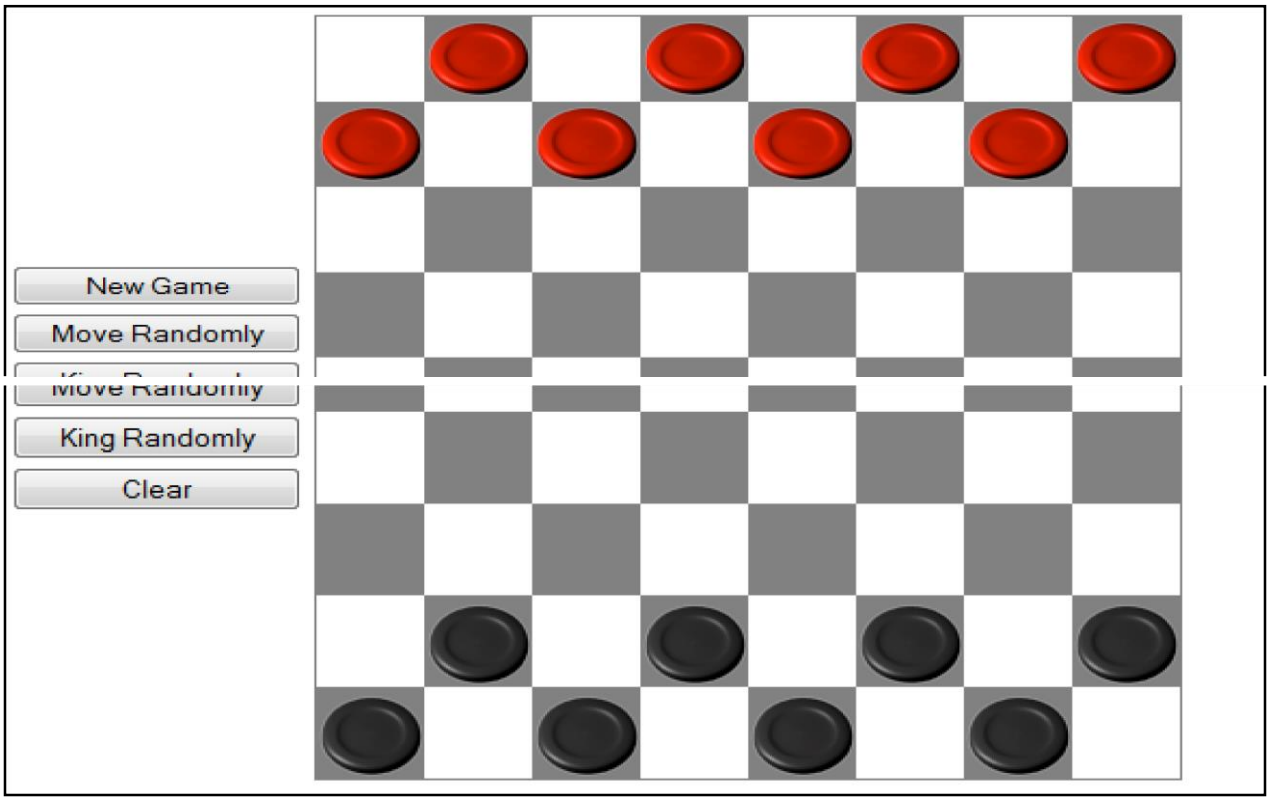
This assignment explores the following topics related to GUI output:

- the object approach;
- the stroke approach (also called vector graphics);
- the pixel approach;
- handling events sent from a model to a view.

In this problem set, you will implement a view that displays a checkers game. You'll implement using a mix of component, stroke, and pixel techniques. In this problem set, you'll only be concerned with output. In the next problem set, you'll add input handling to your views, so that the user can click and drag checkers using the mouse.

You can use HTML or any other IDE that you are comfortable to use in this assignment, and these references may be helpful:

- Here is Mozilla Canvas tutorial



Provided Resources

A zip file containing a main page, stylesheet and javascript libraries for this problem set.

You can import this zip file directly into Eclipse using File/Import/Existing Projects into Workspace, or use whatever text editor you would like. In the root folder of the project are the following files:

- **index.html:** a skeleton file for your user interface
- **mainLayout.css**: a stylesheet file for index.html
- **checker.js**: a javascript file containing the Checker class
- **board.js**: a javascript file containing the Board class
- **boardEvent.js**: a javascript file containing the BoardEvent class
- **graphics**: a folder containing all the graphics files

The board model actually has pieces on it, but you won't see them until you've implemented the pieces display.

Problem 1: Board

Fill in the skeleton of index.html so that it displays a 400x400-pixel checkerboard. The upper left square should be white. The number of squares across the board should be dynamically determined by the
BOARD_SIZE variable in the code. This size defaults to 8x8, but you can change it to any value *n* by adding ?size=*n* to the end of the URL, e.g. index.html?size=16.

No matter how many squares are in the checkerboard, it should always be 400x400 pixels.

You can use either canvas (the stroke approach) or HTML elements (the object approach) to solve this problem.

Problem 2: Checkers

Display all the checkers on the board **using HTML elements** (the object approach). Four pictures are provided for you (red-piece.png, black-piece.png, red-king.png, and black-king.png, found in the graphics folder).

Please don't replace them with different pictures.

Your view must update when the board changes so that it displays the current state of the board at all times. You can test this by clicking on the buttons, which produce various board configurations in the model.

Problem 3: Move Feedback

When a checker moves from one place to another, draw a yellow arrow from the center of its old square to the center of its new square, appearing above all checkers. This arrow should persist until the next change to the board.

Deliverable:

1. **Source code** of your programming assignment *(MUST provide your IDE you are using)* – ***Your completed assignment folder as a zip file that contains all of*** your HTML, JS, and CSS files (including the JSON data), etc.
2. **Screenshots** of your output within a MS Word .doc
3. **MUST Provide a** Short Video Clip
4. ***Submit via d2l Module/dropbox on or before due date***

Here's a checklist of things you should confirm before your submission:

1. Make a fresh folder and unpack your zip file into it
2. Make sure all assets used by your code are found in the fresh folder and load successfully
3. Make sure that the page renders correctly in at least two modern, standards-compliant web browsers (Chrome)