Assignment Requirements

Below are listed the lines in my code in which the corresponding requirements from the assignment are satisfied. Numbering is the same as in the assignment handout.
_____: Criteria _____: Lines in code

1)
Window size = screen size
(index.html) 46: Width and height of canvas equal to browser dimensions (values determined by trial and error).
Viewport area = window size
(main.js) 88-91: Viewport dimensions equal to canvas'.

2)
Black background
(main.js) 414-415: When page is loaded, background set to black.

3)
Number of rectangles
(main.js) 21: Variable 'rectangleNum' is 50.

4)
Colour of each rectangle…
(main.js) 242-246: Array containing colour scheme for each rectangle.
…determined randomly by generating three random numbers…
288: Used random number function for first three arguments (red, green, and blue).
…between 0 and 1…
(main.js) 245: Nothing multiplied or added to random function. Thus numbers are between 0 (inclusive) and 1 (exclusive).
…programmable shader.
(main.js) 161-164: Colour shader.

5)
(main.js) 41-44: One 'tracker' for each rectangle and for each degree of freedom (x, y, and rotation). These trackers contain the state of the given rectangle with respect to the given degree of freedom, and are updated as time elapses.
Initial position of each rectangle determined randomly…
(main.js) 46-50: For each rectangle and for each degree of freedom (x,y, and rotation), assign initial random value. Values are subsequently updated in function 'keepTime' ((main.js) 369-369).
…two integer numbers…
(main.js) 42-43: x and y position trackers.
…between 0 and screen dimensions…
(main.js) 37-38: Maximum absolute x and y position values.
(main.js) 48: Range of absolute values of random numbers is 0 to the corresponding maxima.

6)

Speed determined by 2D vector…

(main.js) 58-59: Speed of translation (contains speed of rotation too), which will be initialized and then remain constant as time elapses.

…initialized with two random numbers…

(main.js) 61-66: For each rectangle and for each degree of freedom (x and y. Rotation also included here), assign random speed.

…between -1.0 and 1.0…

(main.js) 53-54: Maximum x and y speeds.

(main.js) 64: Range of random numbers is -1.0 (ie. 1.0 in the negative direction) to 1.0.

…during animation, this vector is added to initial translation in cumulative manner between each frame.

(main.js) 280: Use matrix translation function on model-view matrix. Translation determined by x and y trackers.

(main.js) 367-371: For each rectangle, updated value of x and y position equal to speed of the given rectangle in the given direction multiplied by elapsed time increment.

7)

Initial size determined randomly…

(main.js) 218-220: For each rectangle, assign random length and width values.

268-274: Assign length and width values to form vertices of a rectangle.

…within some reasonable range.

(main.js) 225-228: Declare maximum and minimum length and width values (these values are distance from the centre point, so actual length and width are twice these values).

(main.js) 218-220: Range of random numbers is minimum length/width to maximum length/width.

8)

Initial orientation expressed as random angle…

(main.js) 39: Maximum absolute angular orientation is 90^0. Because of rectangular symmetry and the fact that the rectangles rotate about their centre, the range $-90 \leq \text{angle} \leq 90$ contains all possible orientations.

(main.js) 44: Third tracker refers to rotational orientation.

(main.js) 45-50: For each rectangle assign initial random angle. Angle values are subsequently updated in function 'keepTime' ((main.js) 381-384).

…direction of rotation determined randomly…

(main.js) 64: Randomly-determined rotational speed has equal probability of being positive or negative.

…increase by randomly generated angular increment…

(main.js) 64: Rotational speed randomly assigned, and remains constant.

(main.js) 283-284: Use matrix rotation function on model-view matrix. Rotation over time determined by rotation tracker.

…between -0.5 and 0.5 degrees…
(main.js) 55: Maximum rotational speed is 0.5 units/ms…
426: …when applied to rotation, this is converted to 45^0/s.
…rotation about centre point of rectangle.
(main.js) 383: Rotation is about 'current' (at a given elapsed time) z-axis. The translation projection matrix shifts ((main.js) 279) these 'current' coordinates to location that serves as origin on which 'current' vertices ((main.js) 225-230) are based. Thus, the 'current' z-axis around which rotation occurs is the centre of the given rectangle.

9)
Rectangles change shape rhythmically…scaling transformation function…
(main.js) 281-282: Use matrix scaling function on model-view matrix, with variable parameters.
…current rotation angle of the rectangle…
(main.js) 281-282: Rotation tracker used as variable in argument of sinusoidal scaling functions.

1)
Double buffering.
Does this refer to position and colour? I used webGL, so maybe this does not matter.

2)
Defined order of transformations.
(main.js) 279-284: Translation, then scaling, then rotation.

3)
Rectangles will remain motionless…
(index.html) 42-43: When program is invoked, scene is set up and drawn in initial state.
…until user presses 's'.
(main.js) 444-446: 's' triggers function call to initiate animation.

4)
Program will quit when user presses 'q'.
(main.js) 447-449: 'q' triggers window to close.

5)
When rectangle exits window…
(main.js) 67-70: 'wrapAround' variables take on values at which wrapping around will occur if a rectangle's centre reaches that value. Based on the size of the screen, the maximum possible dimensions of the rectangle, and the maximum possible instantaneous scaling, wraparound values are the smallest point at which no possible rectangle will still be in view. Thus, most will disappear for several seconds before reappearing, but no disappearances will be visible to the viewer.
(main.js) 372: If a rectangle passes positive boundary…

(main.js) 375: If a rectangle passes negative boundary…
…reappear at complementary position.
(main.js) 373: If a rectangle passes positive boundary, reappears on the positive side of the negative boundary.
(main.js) 376: If a rectangle passes negative boundary, reappears on the negative side of the positive boundary.