CSE 3302/5307 Lab Assignment 4

Due December 8, 2015

Goals:

Understanding of JavaScript prototypal inheritance.

Requirements:

1. Modify the available solution code for Lab 4 Summer 2015 to change the processing for Requirement 3.a. Instead of highlighting (in `"blue"`) the perimeters of all objects containing the cursor position, you should highlight only those objects that *do not contain any other object that includes the cursor position*.

2. Submit your zipped `.html/.js` files on Blackboard by 3:15 p.m. on December 8.

Getting Started:

1. `http://ranger.uta.edu/~weems/NOTES3302/LAB/15SUM/LAB4/` contains the baseline code from Summer 2015 - `lab4.html` and `lab4.js`. The assignment handout is attached.

2. The provided code passes the mouse coordinates to `draw()` methods in the provided code. The decision to draw in `"blue"` is then made locally. In the modified code, you should determine which objects to highlight in `drawing.draw()`. The `insideObject()` methods will be useful.

CSE 3302/5307 Lab Assignment 4

~~Due August 12, 2015~~

Goals:

Understanding of JavaScript prototypal inheritance.

Requirements:

1. Implement a class hierarchy (using `Object.create()`) representing

   a. (1) Simple, convex polygons as a list of counter-clockwise points (last point wraps around to first point).
   b. (2) Rectilinear rectangles as four values: lowX, highX, lowY, highY.
   c. (3) Rectilinear squares as the coordinates of the leftmost lowest point and the length of a side.
   d. (4) A triangle with three counter-clockwise points.
   e. (5) Regular polygon with k counter-clockwise points.

   and including methods to construct an instance, to compute the area, to indicate whether a provided point is inside (including the border) of a region, to indicate whether one object is contained in another, to draw an instance, and ***to clone an object at an offset***.

2. The input is to be taken from an html textarea (in response to a button click) and the output written to an html textarea:

   a. A line indicating the number of objects that follow, one object per line. References to the objects should be placed in a 0-based array.

      1. The number 1, the number of counter-clockwise points, and then pairs of x/y coordinates.
      2. The number 2, followed by the four bounding values.
      3. The number 3, followed by an x/y pair for the leftmost lowest point and then the length of both sides.
      4. The number 4, followed by three x/y pairs for the three counter-clockwise points.
      5. The number 5, followed by the number of sides and then x/y pairs for the first two points. The remaining points will be determined by taking left turns.

   b. A line indicating the number of command lines that follow. The command lines may be formatted as:

      1. The number 1 and an object's subscript to indicate that the area should be computed.
      2. The number 2, an object's subscript, and an x/y pair to be tested for being inside.
      3. The number 3 and one of the five values above for retrieving the number of instances in that class. (a square is a rectangle, while rectangles and triangles are simple polygons; a regular polygon is a simple polygon)
      4. The number 4 and an object's subscript for retrieving the bounding box, a rectilinear rectangle.
      5. The number 5 and two objects' subscripts i and j for testing whether object i is contained in object j. Note that all objects are convex.
      6. ***The number 6, an object's subscript, and an x/y pair for cloning the object by adding the x and y values to all coordinates for the indicated object. A reference to the new object will be included in the existing array. The number of instances in the class hierarchy will also be updated.***

3. After all objects have been created, display all objects in scaled-to-fit form on a canvas. Points should be `"green"`and the perimeter should be `"red"`. For each `mousemove` event, show the new cursor position in a textarea. The displayed coordinates should be based on cartesian coordinates (not window coordinates) and correspond to the displayed objects.

   a. Extend the `mousemove` processing by having the perimeter of all objects containing the cursor position (i.e. by using `inside` methods) turn `"blue"`.

   b. Be sure the perimeter coloring is restored to `"red"` on a `mouseout` event.

4. ~~Submit your zipped `.html/.js` files on Blackboard by 12:45 p.m. on August 12.~~

Getting Started:

1. `http://ranger.uta.edu/~weems/NOTES3302/LAB3SPR15/` contains the baseline code/lab assignment from Spring 2015 - `lab3.html` and `lab3.js`. ***Differences from that assignment are highlighted.***

2. All input values will be provided as strings corresponding to integers. Again, these are conventional cartesian coordinates, not window coordinates.

3. If a polygon is convex, then any three consecutive points will make a left turn.

4. If invalid input is detected, give an error message alert and stop.

5. `http://ranger.uta.edu/~weems/NOTES5311/notes17.pdf` has basic information regarding geometry.

6. Each `clone` method should call the associated `construct` method.

7. Having optional parameters for JavaScript functions can be convenient. If the caller provides fewer arguments (actual parameters) than the number of (formal) parameters, the extra (formal) parameters will have the value `undefined`.