**General Requirements**

* Use the class’ coding conventions (<http://www.w3schools.com/js/js_conventions.asp>)
* Use the GameLoop class you are provided with as the basis for input and output
* Your code needs to run the test scenario as provided in the main.js.
* Name classes, functions and attributes exactly according to the class diagrams when provided
* For fleshing out the “unfinished” classes, use descriptive, meaningful names on each level (except standard names, such as g for Graphics, e for event arguments and I,j.k… for enumeration loops.
* Use small functions that ideally have a single responsibility
* Use small classes that ideally have a single responsibility
* Obey Law of Demeter

**Description of Classes**

|  |  |
| --- | --- |
| ***Class Name*** | **Description** |
| *CustomGameLoop* | Delegates method calls to responsible objects. |
| *ElementManager* | Responsible for keeping collection of Visual objects, drawing, and retrieving visuals at a given position on the canvas |
| *PointerManager* | Manages pointer objects, each representing an independent pointer (e.g. mouse, touch). Establishes connections between pointers and DraggableElements. |
| *Pointer* | Represents a pointer. Keeps position, state (active / inactive) and can be associated with a DraggableElement, which then follows the pointer. |
| *Visual* | Contains all functions and attributes to position, size and draw any visual element to the canvas |
| *HitTestableElement* | Contains all functions necessary for hit testing any elements that have a position and size |
| *DraggableElement* | Encapsulates all attributes and functions necessary for dragging (e.g. start drag, move during drag, end drag). |
| *Shape* | Contains all functions necessary to draw a generic shape |
| *All objects derived from Shape* | Take care only of the individual shape drawing |
| *BackgroundImage* | Load and display within the bounds inherited from Visual |

***Important Notes:***

1. *Even though we accommodate for multi-touch input, you are not expected to test this. If you implement the specified class diagram and specified functionality correctly, it will automatically produce a multi-touch capable solution.*
2. *You can for now assume that there will be only one Pointer per Draggable object (so mutli-touch gestures on a single objects are not part of the specification)*

**Deliverables:**

* Working implementation in Javascript
* Completed UML Diagram that fills in all the attributes and functions for the “empty” objects
* Both elements are to be submitted via GIT

**Grading Criteria:**

* How well does the deliverable run the test scenario?
* Is the deliverable functionally complete?
* Does the deliverable throw exceptions during runtime?
* Does the code contain meaningful names that clearly represent their function and intention of the author?
* Are the classes in the code a) small and b) do they have a clear single responsibility?
* Does the code violate the Law of Demeter?

**Submit by Wed, 10/7, 11:59 PM via GIT (unless you have an extension, everything checked in after the deadline will not be considered).**