# Intro

VASCO[[1]](#footnote-1) is a great tool to demonstrate all kinds of spatial indexing structure and related queries. However, it is built as JAVA applet which is no longer supported by modern browsers due to security problems with the Java browser plug-in. Therefore, a reimplementation of VASCO is needed.

Nowadays, interactive web pages, which is the user interface, are built with JavaScript running on the users’ browser. To avoid constant connections between client and server side, especially during the move operation, data structures and operations are also reimplemented in JavaScript in this project, VASCO-JS.

In the development phase, one file per module is preferred to keep the project clear, which creates lots of JS files. But in the deployment phase, we want users to download minimum JS files to reduce the number of HTTP requests so that pages can be loaded faster. This gap is closed by using RequireJS[[2]](#footnote-2), which keeps tracks of module dependency as well as provides optimization tool for deployment.

# Trees and Operations reimplemented

There are 29 trees in VASCO, 15 for points, 7 for rectangles, 6 for lines and 1 for regions. In VASCO-JS, 8 trees for points and 5 trees for lines are reimplemented.

Table trees reimplemented in VASCO-JS

|  |  |
| --- | --- |
| For points | For lines |
| 1. Point QuadTree 2. K-D Tree 3. PR Quadtree 4. Bucket PR Quadtree 5. PR k-d Tree 6. Bucket PR k-d Tree 7. PMR Quadtree 8. PMR k-d Tree | 1. PM1 Quadtree 2. PM2 Quadtree 3. PM3 Quadtree 4. PMR Quadtree 5. Bucket PM Quadtree |

VASCO provides 7 kinds of operations, such as insert, move, delete, overlap search query, window query nearest neighbor query and within query. In VASCO-JS, the first 3 operations, insert, move and delete, are reimplemented.

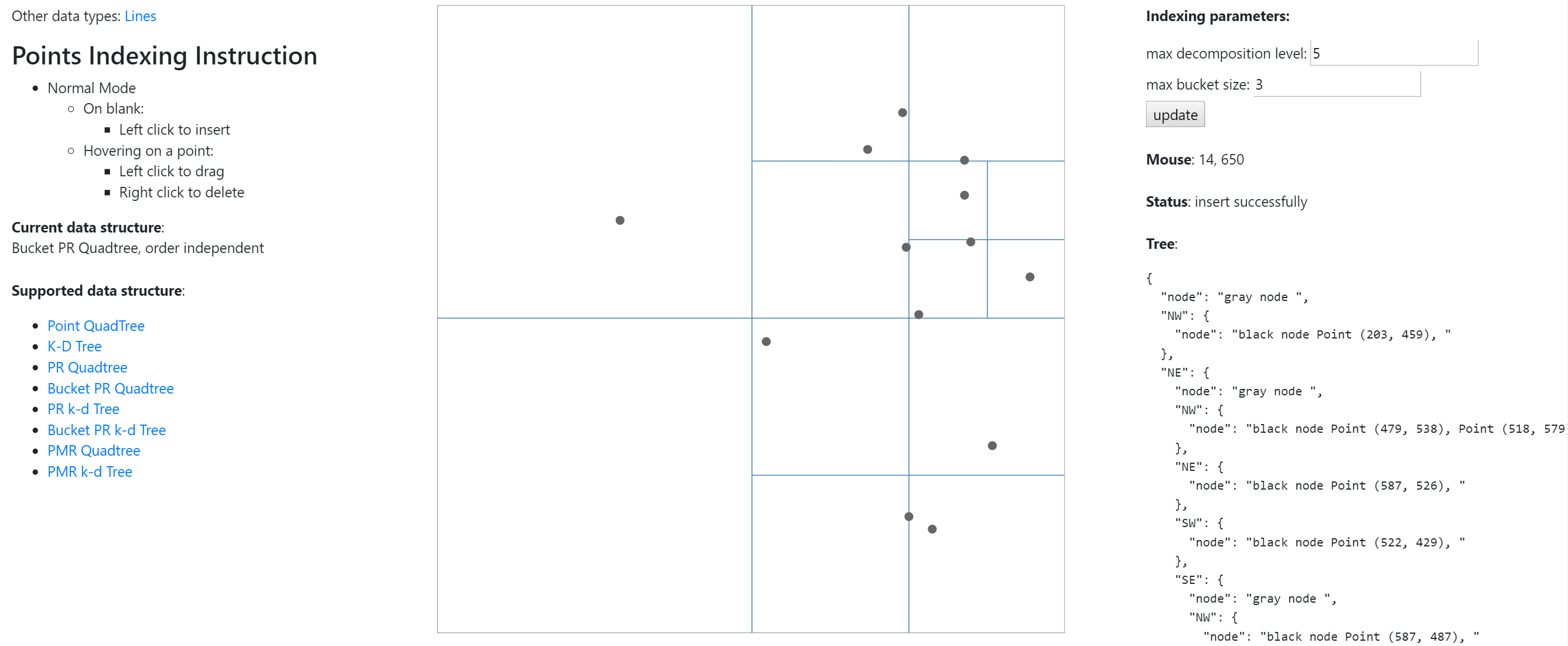
The codes for trees are in /app/indexing folder. Each tree has its own file, \*Tree.js, defining its own insert and delete functions, along with other helper function to perform insert and delete. Then files named as \*TreePub.js are associated with a certain tree implemented public interfaces used by the user interface module.

These public interfaces are:

* init(): init tree structure when loading the user interface or parameters of trees are updated
* toString(): convert the tree structure to string, which is printed on the user interface
* getName(): the name of the current tree structure
* orderDependent(): whether the current tree structure is order dependent
* options(): the input fields containing the default parameters for the current tree structure
* insert(): insert an object into the tree, return false if insertion is rejected, such as exceeding the maximum decomposition level
* del(): delete an object from the tree
* getPartitions(): get space partition from the current tree
* rebuild(): rebuild the tree with current data on the canvas, mostly used during moving operation

# User Interface

To run the applet, open /pt-index.html for points and /ln-index.html for lines.



Above is a snapshot of the user interface using Bucket PR Quadtree for points.

On the left is the instruction of the applet, the name of current data structure, whether it is order dependent and a list of currently available data structures.

On the right are the parameters for the current tree structure, current coordinate of the mouse and data stored in the tree.

In the middle is the canvas for performing operations, insert, move and delete. In VASCO, these are three separate operations meaning you need to explicitly choose the operation mode. Here in VASCO-JS, these three operations can be done simultaneously. Insertion is performed when you click on the blank of the canvas. You can snap an endpoint of a line to another line by moving one of the endpoints closed to another. If the insertion is rejected by the tree structure, the new object will be discarded. You can delete an object by right click on it and move on an object by pressing left click and drag it. The advantage of this interaction is that it is intuitive, and you don’t need to keep switching modes. But the downside is you need to move your mouse exactly on the object, which is sometimes not so easy as the object is small.

1. http://www.cs.umd.edu/ hjs/quadtree/index.html [↑](#footnote-ref-1)
2. http://requirejs.org/ [↑](#footnote-ref-2)