Public API

Edit New Page

Jump to bottom

Sébastien Heymann edited this page on Feb 15, 2015 · 13 revisions

The object `sigma` is at the same time the constructor and the package containing every sigma related objects, classes and functions.

Constructor

Basically, sigma must be initialized with a **configuration object** as argument. Almost everything recognized in this object can be updated after instantiation with the provided public methods. But there are some shortcuts that are recognized by the constructor to simplify the most common use cases:

- If no parameter is given to the constructor, the instance will be created without any renderer or camera. It will just instantiate the graph, and other modules will have to be instantiated through the public methods, like `addRenderer` for instance:

```
var s = new sigma();
s.addRenderer({
  type: 'canvas',
  container: 'my-container-id'
});
```

- In most of the cases, sigma will simply be used with the default renderer. Then, since the only required parameter is the **DOM container**, there are some simpler way to call the constructor. The four following calls do the exact same things:

```
var s1 = new sigma('my-container-id');
var s2 = new sigma(document.getElementById('my-container-id'));
var s3 = new sigma({
  container: document.getElementById('my-container-id')
});
var s4 = new sigma({
  renderers: [{
    container: document.getElementById('my-container-id')
  }]
});
```

Here is an exhaustive list of the **parameters that are recognized in the configuration object**:

- **id** : *string*
  - The ID of the instance. It will be generated automatically if not specified, from 0 and getting incremented at each new instance. It is possible to retrieve any instance of sigma with the static method `sigma.instances`. And finally, if sigma is instantiated with the ID of an already existing instance, it will throw an error. Finally, killing an instance with its method `kill` will effectively dereference its ID.
- **renderers** : *array*
  - An array containing objects describing renderers.
- **graph** : *object*
  - An object containing an array of nodes and an array of edges, to avoid having to add them by hand later. Also, if and only if a graph is given through the configuration object, the method `refresh` of the sigma instance will be called directly from the controller, to keep you from having to call it explicitly. The object will be entered in the graph through the method `read` of `sigma.classes.graph`. Check the graph documentation for more information.
- **settings** : *object*
  - An object containing settings that will override the default ones defined in the object `sigma.settings`, but only for **this instance**. Check the settings documentation for more information.

Public methods

Once the sigma instance is constructed, it will provide several methods to manipulate cameras and renderers, as well as to force rendering of data processing. First of all, it inherits from the classes `sigma.classes.dispatcher` and `sigma.classes.configurable` (documentation here).

Cameras and renderers manipulation

- **addCamera(?*string*)** : *camera*
  - This methods will instantiate and reference a new camera. Basically, it will bind the render methods to the camera's events, and initialize the camera's own quadtree to find which nodes are on screen and some objects required to deal with the camera
  - If no `id` is specified, then an automatic `id` will be generated.
- **killCamera(*string*|*camera*)** : *sigma*
  - This method kills a camera, and every renderer attached to it. It is possible to call this method with a camera or its `id`.
  - It returns the sigma instance.
- **addRenderer(?*object*)** : *renderer*

    - This methods will instantiate and reference a new renderer. The argument can be the constructor or its name in the `sigma.renderers` package (`"canvas"` or `"webgl"` , for instance). If no type is specified, then `sigma.renderers.def` will be used instead.
    - If no `id` is specified, then an automatic id will be generated.
- **killRenderer(*string|camera*)** : *sigma*
    - This method dereferences and kills a renderer. If the renderer has a `kill` method, then it will be called.
    - It returns the sigma instance.
- **kill()**
    - This method calls the `kill` method of each module that has one, and destroys any reference from the instance.

Rendering methods

The following methods deal with actual rendering. They are the methods used inside or outside sigma to order the renderers to render the graph.

These methods are multiple, since the complexity of sigma requires some different way to render for different usages.

- **renderCamera(*camera*, ?*boolean*)** : *sigma*
    - This method calls the `render` method of each renderer that is bound to the specified camera. To improve the performances, if this method is called too often, the number of effective renderings is limited to one per frame, unless the second argument evaluates to 'true'.
    - It has been initially designed to render only one camera when the user drags the graph or zoom in the graph, there should not be many cases where you might use it by yourself from outside sigma.
    - It returns the sigma instance.
- **render()** : *sigma*
    - This method calls the `render` method of each renderer.
    - It has been initially designed to render every renderers from inside sigma, but it actually should not be used from outside most of the time.
    - It returns the sigma instance.
- **refresh(?*options*)** : *sigma*
    - This method calls the `render` method of each renderer, with the same arguments, but will also check if the renderer has a `process` method, and call it if it exists. The method will trigger the spatial indexing of the graph. As this is a costly process, you may set the option `skipIndexation: true` to refresh without updating the spatial index.
    - **This is the method to call if you have changed anything in the graph**. And it is very important: For instance, any modification in the graph (even the most little, like one color or one label) will be observed in the WebGL renderer only after the `refresh` method has been called.
    - It returns the sigma instance.

+ Add a custom footer

Pages 8

Find a Page...

- Home
- Camera API
- Events API
- Graph API
- Public API
- Renderers
- Settings
- Settings: How Settings Work in Sigma

+ Add a custom sidebar

Clone this wiki locally

https://github.com/jacomyal/sigma.js.wiki.git