UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL DE FI DE GRAU

Grau en Enginyeria Biomèdica

DEVELOPMENT OF A WEB-BASED GRAPHICAL USER INTERFACE TO DESIGN BRAIN FIBER MODELS FOR TRACTOGRAPHY VALIDATION

Volum II

Annex

Autor: Guillem González Vela
Director: Jordi Solà Soler
Departament ESAII
Co-Director: Emmanuel Caruyer

Convocatòria: Juny de 2017

Contents

Chapter 1

User Documentation

User Documentation

Phantomαs Web Designer

Author: Guillem González Vela, Emmanuel Caruyer - Firstname.Lastname@irisa.fr

Source: Phantomαs Web Designer is on GitHub

Phantomαs Web Designer is a graphical interface for creation and edition of phantoms to be used in Phantomαs (link to Phantomαs' homepage).

Contents

1 Requirements

Phantomαs Web Designer was tested on Mozilla Firefox. Although, it is fully compatible with any modern internet navigator.

No extra software is needed.

2 Capabilities

Using this app you will be able to

- Load or create from scratch any phantom model and save it as a JSON Phantomαs file

- Visualize any phantom model and its structure and components individually in a lightweight, three-dimensional and fully interactive interface.
- Add and remove fibers and isotropic regions in the phantom.
- Change the position and the radius for any isotropic region
- Edit the radius and the tangents' method for any fiber
- Add, remove and edit the position of any control point in a fiber

2.1 Definition of a fiber bundle

A fiber bundle in Phantomαs is defined as a cylindrical tube wrapped around its centerline. The centerline itself is a continuous curve in 3D, and can be simply created from a few control points.

This app uses the same specifications as Phantomαs. More information may be found in its documentation.

3 Basic Usage

Phantomαs Web Designer divides the window in three panels:

- Left panel allows you to navigate through the elements and edit them
- Central panel displays the fiber in a what-you-see-is-what-you-get manner. It is fully mouse interactive.
- Right panel lets the user tweak the display options.

3.1 Phantom overview

The largest panel displays the current phantom layout. At start, view is from plane XY.

You may click to rotate, right-click to pan and use the mouse wheel to zoom. The view may be restored at any time by using the right panel camera placements.

To identify and view the structure of individual phantom elements, you may use the left panel element navigator.

3.2 Left panel

The left panel allows you to navigate between the different phantom elements. You may also edit those, or add and remove.

To identify the different elements you may place your mouse over the selector lists. Those will highlight while mouse is placed onto their selection option. Highlighting elements for identifying will not affect the current task.

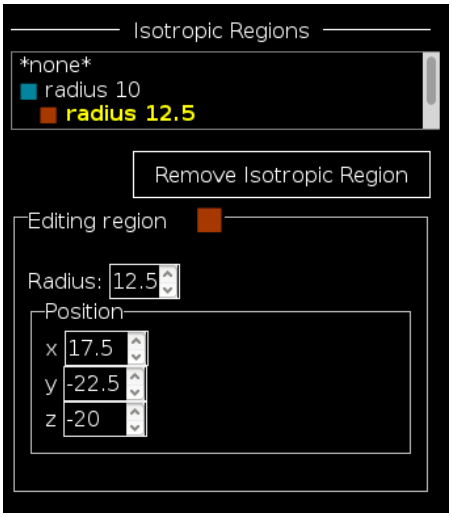
3.2.1 Edition mode

When clicking onto an option, edition options will pop up and the element will stay highlighted. To exit edition mode, select the **none** option or press *Esc*. Changes are saved once those take place.

To remove an element, you must access its edition mode.

Numeral inputs allow keyboard input, although it is restricted to valid values and a 1-decimal precision by default.

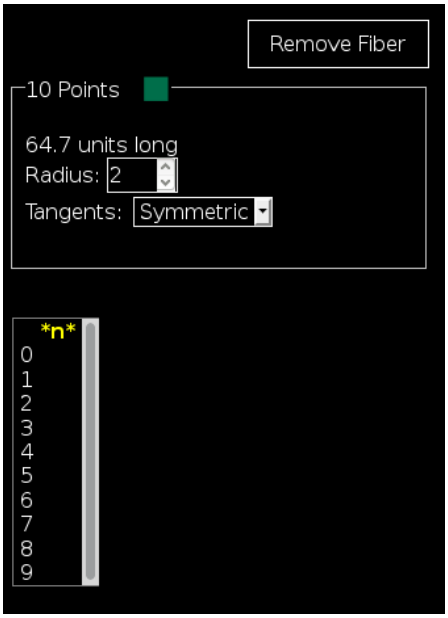
3.2.2 Editing an Isotropic Region



The editable elements in a region concern:

- Its radius
- Its position (x y z)

3.2.3 Editing a Fiber



When entering fiber edition mode, its structure will feature in the scene.

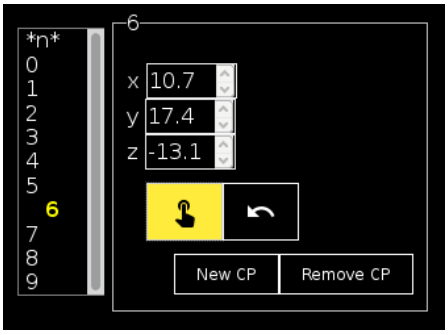
The editable elements in a fiber concern:

- Its radius
- Its tangent-computing mode:
  - Symmetric
  - Incoming
  - Outgoing

A selection list for the fibers' control points is available at the bottom. Hovering those will also highglight them in the scene to help identifying.

3.2.4 Editing a Control Point

Clicking over a control point in the list will pop up control point edition mode.



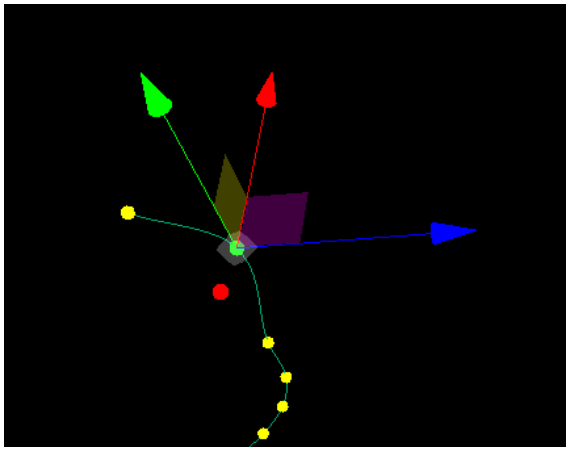
Only position may be edited.

While editing and navigating through control points, those may adopt four different colors in the scene:

- *Yellow*: Control points the user is not interacting with.
- *Blue*: For the control point being hovered in selection list.
- *Red*: Currently being edited control point.
- *Green*: Position changed control point that may be saved.

A control point is saved whenever its edition is quitted. When position was changed, the former version is on the scene in red color while the one to be saved is in green. The former may be recovered by pressing the *Undo* button, marked with an arrow.

Position may be edited manually by using the given fields. As well, by clicking on the *Drag and Drop* button, marked with a pointer. This option will disable the manual fields while allowing an interactive edition in the scene itself.



The interactive edition allows different liberty grades:

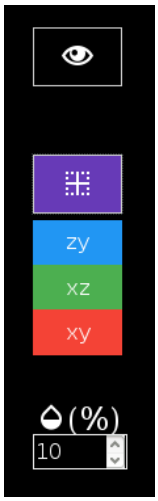
- *Axes*, by using the arrows shown.
- *Planes*, by dragging the planes formed between the arrows.
- *The screen plane*, by clicking directly on the point.

New CP button creates a new control point. It is to be placed in the mid-point between the current and the next one. *Remove CP* removes the current control point after asking for confirmation.

Remove CP option is not available in first and last control points.

3.3 Right panel

Right panel concerns everything regarding the visualization.



Options are:

- *Preview Switch*: Only available in editing mode. Allows the user to preview the phantom with any fade. Does not affect the edition. Unactive in the screenshot above.
- *Axes Switch*: Shows and hides the coordinates axis. Active in the screenshot above. Each axis features a different color:
  - Red for X
  - Green for Y
  - Blue for Z
- *Position*: Move the scene to XY, XZ or ZY plane.
- *Opacity*: Select [0 - 50%] the opacity of the faded elements.

3.4 Export Phantom

At bottom-right corner the export button allows you, at any moment, to download the JSON file for the current phantom.

The file is compatible with Phantomαs and may be loaded as well in Phantomαs Web Designer for further edition.

3.5 Keyboard Shortcuts

Keyboard shortcuts are available for most commonly used functions:

Esc	Exit current edit
P	Switch preview mode
D	Switch drag and drop controls
A	Switch axes
X	Move to X=0 plane
Y	Move to Y=0 plane
Z	Move to Z=0 plane
S	Save phantom
U	Undo control point edition
Del	Remove current element

4 Source Code

Phantomαs Web Designer is open source and may be downloaded and forked on GitHub. Pull Requests are welcome!

This document was generated on 2017-04-27 at 11:13.

Chapter 2

Source Code

2.1 File tree

```
phantomas-web/
├── index.html
├── phantomas.html
├── css/
│   ├── icons.css
│   ├── main.css
│   ├── normalize.css
│   └── w3.css
├── doc/
│   ├── jsdoc-conf.json
│   ├── developer/
│   │   └── index.html
│   └── user/
│       ├── img/
│       ├── source.rst
│       └── index.html
├── examples/
│   ├── 3Dfanning_13bundles.json
│   ├── 60crossing_3bundles.json
│   ├── 90kissing_3bundles.json
│   ├── fibers.json
│   └── isbi_challenge_2013.json
├── gui/
│   ├── cpedit.js
│   ├── fiberedit.js
│   ├── handlers.js
│   ├── regionedit.js
│   ├── resize.js
│   ├── setup.js
│   ├── status.js
│   └── stylehandlers.js
├── icons/
│   ├── icons.woff
│   └── favicon.ico
├── js/
│   ├── FiberSource.js
│   ├── MeshSource.js
│   ├── axes.js
│   ├── load.js
│   ├── main.js
│   ├── Phantom.js
│   ├── save.js
│   └── dragAndDrop.js
└── lib/
    ├── TrackballControls.js
    ├── three.min.js
    └── TransformControls.js
```

2.2 File description

File name	Description	Page
`index.html`	*Phantomαs* Web Designer's homepage	14
`phantomas.html`	*Phantomαs* Web Designer's HTML	15
`icons.css`	Class definition for button icons	17
`main.css`	Main classes definition	17
`jsdoc-conf.json`	*JSDoc3* configuration file	19
`source.rst`	User documentation reStructuredText source	20
`cpedit.js`	`cpEdit` and `exitCPedit` functions	25
`fiberedit.js`	`fiberEdit` function	29
`handlers.js`	GUI Handlers functions	33
`regionedit.js`	`regionEdit` function	40
`resize.js`	`resizeGUI` function	43
`setup.js`	`guiSetup` and `editExit` functions	44
`status.js`	GuiStatus class	49
`stylehandlers.js`	GUI Style Handlers functions	51
`FiberSource.js`	`FiberSource` and `IsotropicRegionSource` classes	52
`MeshSource.js`	Mesh-wrapper classes	60
`axes.js`	`buildAxes` function	65
`load.js`	`loadPhantom` function	66
`main.js`	Variable declaration. `init` and `show` functions	67
`Phantom.js`	`Phantom` class	71
`save.js`	`Phantom.export` method, `pushDownload` function	82
`dragAndDrop.js`	`dragAndDrop` function	85

2.3 Source code

2.3.1 index.html

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <link rel="icon" href="icons/favicon.ico">
    <meta name="author" content="Guillem Gonzalez Vela, Emmanuel Caruyer - Firstname.Lastname&#64;irisa.fr" />
    <title>Phantom&alpha;s Web</title>
    <center><h1>Phantom&alpha;s Web</h1></center>
    <hr>
    <br><br><br>
  </head>

  <body style="background: black; color: white">
    <center><table style="padding:7%; width:80%; background: white; color: black"></center>
      <tr><center>
        <th><h2>Load the app</h2></th>
        <th><h2>Other resources</h2></th>
      </center></tr>
      <tr>
        <td>
          <h3><a href="phantomas.html">Start a new model from scratch</a></h3>
          <h4>or load an example:</h4>
          <ul>
            <li><a href="phantomas.html?examples/fibers.json">Basic phantom</a>
            <li><a href="phantomas.html?examples/3Dfanning_13bundles.json">13 fanning bundles</a>
            <li><a href="phantomas.html?examples/60crossing_3bundles.json">3 bundles crossing at 60&ordm;</a>
            <li><a href="phantomas.html?examples/90kissing_3bundles.json">3 bundles kissing at 90&ordm;</a>
            <li><a href="phantomas.html?examples/isbi_challenge_2013.json">Phantom used at 2013 HARDI reconstruction challenge</a>
          </ul>
        </td>
        <td style="padding-left:10%">
          <b>Authors:</b> Guillem Gonz&aacute;lez Vela and Emmanuel Caruyer (Firstname.Lastname@irisa.fr)
          <br><br>
          <ul>
            <li><a href="doc/user">User documentation</a>
            <li><a href="doc/developer">Developer documentation</a>
            <li><a href="LICENSE">License - BSD 2-Clause License<a>
          </ul>
          <br>
          External links:
          <ul>
```

```
42            <li>This app is Open Source and
43              <a href="http://www.github.com/ecaruyer/phantomas-web"
                    target="_blank">may be forked in GitHub.<a> Pull
                    requests are welcome!
44            <li><a href="http://emmanuelcaruyer.com/phantomas.php" target=
                  "_blank">Phantom&alpha;s' homepage</a>
45            <li><a href="http://www.github.com/ecaruyer/phantomas" target=
                  "_blank">Phantom&alpha;s' GitHub</a>
46            <li><a href="http://emmanuelcaruyer.com/phantomas/" target="
                  _blank">Phantom&alpha;s' API Documentation</a>
47            <li><a href="http://hardi.epfl.ch/static/events/2013_ISBI/"
                    target="_blank">HARDI reconstruction challenge 2013
                    homepage</a>
48          </ul>
49    </body>
50  </html>
```

2.3.2 phantomas.html

```
1   <!DOCTYPE html>
2   <html>
3
4   <head>
5     <title>Phantomas Web Designer</title>
6     <meta charset="utf-8">
7     <link rel="icon" href="icons/favicon.ico">
8     <link rel="stylesheet" href="css/normalize.css" />
9     <link rel="stylesheet" href="css/w3.css" />
10    <link rel="stylesheet" href="css/main.css" />
11    <link rel="stylesheet" href="css/icons.css">
12
13    <script type="text/javascript" src="lib/three.min.js"></script>
14    <script type="text/javascript" src="lib/TrackballControls.js"></script>
15    <script type="text/javascript" src="lib/TransformControls.js"></script>
16
17    <script type="text/javascript" src="js/FiberSource.js"></script>
18    <script type="text/javascript" src="js/MeshSource.js"></script>
19    <script type="text/javascript" src="js/Phantom.js"></script>
20    <script type="text/javascript" src="js/dragAndDrop.js"></script>
21    <script type="text/javascript" src="js/load.js"></script>
22    <script type="text/javascript" src="js/save.js"></script>
23    <script type="text/javascript" src="js/axes.js"></script>
24    <script type="text/javascript" src="js/main.js"></script>
25
26    <script type="text/javascript" src="gui/handlers.js"></script>
27    <script type="text/javascript" src="gui/resize.js"></script>
28    <script type="text/javascript" src="gui/status.js"></script>
29    <script type="text/javascript" src="gui/setup.js"></script>
30    <script type="text/javascript" src="gui/stylehandlers.js"></script>
31    <script type="text/javascript" src="gui/fiberedit.js"></script>
32    <script type="text/javascript" src="gui/regionedit.js"></script>
33    <script type="text/javascript" src="gui/cpedit.js"></script>
34  </head>
35
```

```
36 <body>
37   <div id="leftGUI">
38     <!-- Title code from http://stackoverflow.com/questions/2812770/add-
         centered-text-to-the-middle-of-a-hr-like-line -->
39     <div style="width: 100%; height: 8px; border-bottom: 1px solid white;
         text-align: center">
40       <span style="background-color: #000000; padding: 0 10px">
41         Fibers
42       </span>
43     </div>
44     <ul id="fiberSelector" style="width:120px;" onmouseout="guiStatus.
         retrieve();">
45     </ul><br>
46
47     <div style="width: 100%; height: 8px; border-bottom: 1px solid white;
         text-align: center">
48       <span style="background-color: #000000; padding: 0 10px">
49         Isotropic Regions
50       </span>
51     </div>
52     <ul id="regionSelector" style="width:120px" onmouseout="guiStatus.
         retrieve();">
53     </ul>
54
55     <div id="editGUI">
56     </div>
57
58   </div>
59
60   <div id="container">
61   </div>
62
63   <div id="rightGUI">
64     <button id='switchViewButton' title="Preview (P)" class='w3-btn w3-
         border w3-hover-aqua w3-block w3-ripple' onclick="switchViewButton
         ();">
65     <i class="icons">&#xE9CE;</i></button>
66     <br><br>
67
68     <ul>
69       <li>
70         <button id='toggleAxesButton' title="Axes (A)" style='margin-
             bottom: 5px' class='w3-btn w3-border w3-hover-deep-purple w3-
             block w3-ripple' onclick='toggleAxes()'>
71           <i class="icons">&#xE22D;</i></button>
72       </li>
73       <li>
74         <input type='button' class="w3-button w3-blue w3-block" value='zy'
              title="X Plane (X)" onclick='moveCameraZY();'>
75       </li>
76       <li>
77         <input type='button' class="w3-button w3-green w3-block" value='xz
             ' title="Y Plane (Y)" onclick='moveCameraXZ();'>
78       </li>
```

```
79        <li>
80          <input type='button' class="w3-button w3-red w3-block" value='xy'
                title="Z Plane (Z)" onclick='moveCameraXY();'>
81        </li>
82      </ul>
83
84      <br>
85      <label>
86      <i class="icons">&#xE91C;<span style='font-family:default;'>(%)</span
            ></i>
87      </label>
88      <input id='opacitySelector' style='width:100%' type="number" min='0'
            max='50' step='5' onchange='opacitySelectChange(this);' />
89
90      <div id="bottomButtons">
91        <button class="w3-button w3-grey w3-hover-green w3-ripple w3-block"
              onclick="saveClick();" title="Save (S)">
92          <i class="icons">&#xE9C7;</i></button>
93      </div>
94    </div>
95
96    <a id="downloadAnchorElem" style="display:none">
97      <!-- Must stay empty, to be used for pushing phantom download -->
98    </a>
99
100 </body>
101
102 </html>
```

2.3.3 icons.css

```
1  @font-face {
2    font-family: 'appicons';
3    font-style: normal;
4    font-weight: 400;
5    src: url(../icons/icons.woff) format('woff');
6  }
7
8  .icons {
9    font-family: 'appicons';
10   font-weight: normal;
11   font-style: normal;
12   font-size: 24px;
13   line-height: 1;
14   letter-spacing: normal;
15   text-transform: none;
16   display: inline-block;
17   white-space: nowrap;
18   word-wrap: normal;
19   direction: ltr;
20 }
```

2.3.4 main.css

```
1    hml, html, body {
```

```
2     margin: 0;
3     padding: 0;
4     color: white;
5     background-color: black;
6   }
7
8   input[type=number], select {
9     background-color: black;
10    color: white;
11    border-width: thin;
12    border-style: solid;
13    height: 23px;
14  }
15
16  ul, li {
17      list-style: none;
18      padding-left:0;
19      color: white;
20  }
21
22  .enabledList {
23    overflow: scroll;
24    overflow-x: hidden;
25    cursor: pointer;
26    padding-left: 5px;
27
28    border-style: solid;
29    border-width: 1px;
30    border-color: grey;
31    border-right: 0;
32  }
33  .disabledList {
34    background-color: grey;
35    padding-left: 10px;
36  }
37
38  .optionUnselected {
39  }
40  .optionSelected {
41    padding-left: 15px;
42    color:yellow;
43    font-weight:bold;
44  }
45  .optionOnMouseOver {
46    font-weight:bold;
47    background-color: DimGrey;
48  }
49  .optionSelectedAndOnMouseOver {
50    background-color: DimGrey;
51    padding-left: 20px;
52    color:yellow;
53    font-weight:bold;
54
55    /*font-style: oblique;*/
```

```
56   }
57
58   #leftGUI {
59     float: left;
60     text-align: left;
61     display: inline-block;
62
63     width: 19%;
64     margin: 0;
65     padding: .5%;
66   }
67
68   #container {
69     display: inline-block;
70     margin: 0;
71     padding: 0;
72
73     width: 73%;
74     height: 99%;
75   }
76
77   #rightGUI {
78     float: right;
79     text-align: center;
80     display: inline-block;
81
82     width: 4%;
83
84     margin: 0;
85     padding: .5%;
86   }
87
88   #bottomButtons {
89     position: absolute;
90     bottom: 0;
91     right: 0;
92
93     width: 4%;
94
95     padding: .5%;
96     padding-left: 0;
97   }
```

2.3.5 jsdoc-conf.json

```
1  {
2    "source": {
3      "include": ["js", "gui"],
4      "includePattern": ".+\\.js(doc|x)?$",
5      "excludePattern": "(^|\\/|\\\\)_"
6    },
7    "opts": {
8      "destination": "./doc/developer/"
```

```
  },
  "tags": {
    "allowUnknownTags": true,
    "dictionaries": ["jsdoc","closure"]
  },
  "plugins": [],
  "templates": {
      "cleverLinks": true,
      "monospaceLinks": false
  }
}
```

2.3.6 source.rst

```
.. -*- coding: utf-8 -*-
.. raw:: html

  <link rel="icon" href="../../icons/favicon.ico">


==========================================
User Documentation
==========================================
------------------------------------------
Phantom |alpha| s Web Designer
------------------------------------------

:Author: Guillem Gonzalez Vela, Emmanuel Caruyer - Firstname.Lastname\
    @irisa.fr
:License: ‘BSD 2-Clause License‘_
:Source: Phantom |alpha| s Web Designer is on GitHub_

.. _BSD 2-Clause License: ../../LICENSE
.. _GitHub: https://github.com/ecaruyer/phantomas-web

Phantom |alpha| s Web Designer is a graphical interface for creation and
edition of phantoms
to be used in Phantom |alpha| s (‘link‘_ to Phantom |alpha| s’ homepage).

.. _link: http://www.emmanuelcaruyer.com/phantomas.php

.. contents::
.. section-numbering::


Requirements
====================
Phantom |alpha| s Web Designer was tested on
‘Mozilla Firefox‘_. Although, it is fully compatible
with any modern internet navigator.

No extra software is needed.
```

```
38
39 .. _Mozilla Firefox: http://www.firefox.com
40
41
42 Capabilities
43 ====================
44 Using this app you will be able to
45
46   - Load or create from scratch any phantom model and save it as a
47     JSON Phantom |alpha| s file
48   - Visualize any phantom model and its structure and components
         individually
49     in a lightweight, three-dimensional and fully interactive interface.
50   - Add and remove fibers and isotropic regions in the phantom.
51   - Change the position and the radius for any isotropic region
52   - Edit the radius and the tangents' method for any fiber
53
54   - Add, remove and edit the position of any control point in a fiber
55
56 Definition of a fiber bundle
57 ----------------------------
58
59 A fiber bundle in Phantom |alpha| s is defined as a cylindrical tube
       wrapped around
60 its centerline. The centerline itself is a continuous curve in
61 3D, and can be simply created from a few control points.
62
63 This app uses the same specifications as Phantom |alpha| s. More
       information
64 may be found in its documentation_.
65
66 .. _documentation: http://www.emmanuelcaruyer.com/phantomas/fiber_bundle.
       html
67
68
69 Basic Usage
70 ====================
71 .. contents:: :local:
72
73 Phantom |alpha| s Web Designer divides the window in three panels:
74
75 - Left panel allows you to navigate through the elements and edit them
76 - Central panel displays the fiber in a what-you-see-is-what-you-get
       manner.
77   It is fully mouse interactive.
78
79 - Right panel lets the user tweak the display options.
80
81 .. image:: img/capture.png
82     :width: 60%
83     :align: center
84
85 Phantom overview
86 ----------------------
```

```
87  The largest panel displays the current phantom layout. At start, view is
        from
88  plane XY.
89
90  You may click to rotate, right-click to pan and use the mouse wheel to
        zoom.
91  The view may be restored at any time by using the ‘right panel‘_
92  camera placements.
93
94  To identify and view the structure of individual phantom elements, you may
         use
95  the ‘left panel‘_ element navigator.
96
97  Left panel
98  ----------------------
99  The left panel allows you to navigate between the different phantom
        elements.
100 You may also edit those, or add and remove.
101
102 To identify the different elements you may place your mouse over the
        selector
103 lists. Those will highlight while mouse is placed onto their selection
        option.
104 Highlighting elements for identifying will not affect the current task.
105
106 Edition mode
107 ‘‘‘‘‘‘‘‘‘‘‘‘
108 When clicking onto an option, edition options will pop up and the element
        will
109 stay highlighted. To exit edition mode, select the *\*none\** option or
        press
110 *Esc*. Changes are saved once those take place.
111
112 To remove an element, you must access its edition mode.
113
114 Numeral inputs allow keyboard input, although it is restricted to valid
        values
115 and a 1-decimal precision by default.
116
117 Editing an Isotropic Region
118 ‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
119 .. image:: img/regionedit.png
120     :align: center
121
122 The editable elements in a region concern:
123
124   - Its radius
125
126   - Its position (x y z)
127
128 Editing a Fiber
129 ‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
130 .. image:: img/fiberedit.png
131     :align: center
```

```
132
133 When entering fiber edition mode, its structure will feature in the scene.
134
135 The editable elements in a fiber concern:
136
137   - Its radius
138
139   - Its tangent-computing mode:
140
141     + Symmetric
142     + Incoming
143
144     + Outgoing
145
146 A selection list for the fibers' control points is available
147 at the bottom.
148 Hovering those will also highglight them in the scene to help identifying.
149
150 Editing a Control Point
151 ''''''''''''''''''''''''''''''''''''''''
152 Clicking over a control point in the list will pop up control
153 point edition mode.
154
155 .. image:: img/cpedit.png
156     :align: center
157
158 Only position may be edited.
159
160 While editing and navigating through control points, those may adopt four
161 different colors in the scene:
162
163   - *Yellow*: Control points the user is not interacting with.
164   - *Blue*: For the control point being hovered in selection list.
165   - *Red*: Currently being edited control point.
166
167   - *Green*: Position changed control point that may be saved.
168
169 A control point is saved whenever its edition is quitted. When position
        was
170 changed, the former version is on the scene in red color while the one to
171 be saved is in green. The former may be recovered by pressing the *Undo*
        button,
172 marked with an arrow.
173
174 Position may be edited manually by using the given fields. As well, by
        clicking
175 on the *Drag and Drop* button, marked with a pointer. This option will
        disable
176 the manual fields while allowing an interactive edition in the scene
        itself.
177
178 .. image:: img/draganddrop.png
179     :align: center
180
```

```
The interactive edition allows different liberty grades:

  - *Axes*, by using the arrows shown.
  - *Planes*, by dragging the planes formed between the arrows.

  - *The screen plane*, by clicking directly on the point.

*New CP* button creates a new control point. It is to be
placed in the mid-point between the current and the next one. *Remove CP*
removes the current control point after asking for confirmation.

*Remove CP* option is not available in first and last control points.

Right panel
----------------------
Right panel concerns everything regarding the visualization.

.. image:: img/rightpanel.png
    :align: center

Options are:

  - *Preview Switch*: Only available in editing mode.
    Allows the user to preview the phantom with any fade. Does not affect
    the edition. Unactive in the screenshot above.
  - *Axes Switch*: Shows and hides the coordinates axis. Active
    in the screenshot above. Each axis features a different color:

    + Red for X
    + Green for Y
    + Blue for Z

  - *Position*: Move the scene to XY, XZ or ZY plane.

  - *Opacity*: Select [0 - 50%] the opacity of the faded elements.

Export Phantom
----------------------
At bottom-right corner the export button allows you, at any moment, to
    download
the JSON file for the current phantom.

The file is compatible with Phantom |alpha| s and may be loaded as well in
     Phantom |alpha| s
Web Designer for further edition.

Keyboard Shortcuts
----------------------
Keyboard shortcuts are available for most commonly used functions:

====  ============================================
Esc    Exit current edit
----  --------------------------------------------
P      Switch preview mode
```

```
233 ----  ------------------------------------------
234 D      Switch drag and drop controls
235 ----  ------------------------------------------
236 A      Switch axes
237 ----  ------------------------------------------
238 X      Move to X=0 plane
239 ----  ------------------------------------------
240 Y      Move to Y=0 plane
241 ----  ------------------------------------------
242 Z      Move to Z=0 plane
243 ----  ------------------------------------------
244 S      Save phantom
245 ----  ------------------------------------------
246 U      Undo control point edition
247 ----  ------------------------------------------
248 Del    Remove current element
249 ====  ==========================================
250
251 Source Code
252 ===========================
253 Phantom |alpha| s Web Designer is open source and may be
254 'downloaded and forked on GitHub'_. Pull Requests are welcome!
255
256 .. _downloaded and forked on GitHub: https://github.com/ecaruyer/phantomas
        -web
257
258
259 .. raw:: html
260
261   <center><br><br><br>
262
263 ----------------------------------------
264
265 This document was generated on |date| at |time|.
266
267
268
269 .. |alpha| unicode:: U+03B1 .. alpha
270   :trim:
271
272 .. |date| date::
273 .. |time| date:: %H:%M
```

2.3.7 cpedit.js

```
1 /**@overview Code creating and removing the Control Point edition UI*/
2
3 function cpEdit(index) {
4 /** @function cpEdit
5   * @memberof module:GUI Construction
6   * @desc Constructs the Control Point edition UI for a given index of a
        control point.
7   * @param {Number} index The index of the Control Point to edit.
8   */
```

```
9   var fiber = phantom.fibers.source[guiStatus.editingFiber];
10  var cp = fiber.controlPoints[index];
11  var former = guiStatus.formerCP;
12  var cpEditor = document.getElementById("cpEditor");
13
14  // The edit field is appended to the second CPedit table
15  var field = document.createElement("FIELDSET");
16  field.id = "cpEditField";
17  cpEditor.innerHTML = "";
18  cpEditor.appendChild(field);
19
20  var title = document.createElement("LEGEND");
21  title.innerHTML = ' ' + index.toString(); + ' ';
22  field.appendChild(title);
23
24  // POSITION PROPERTIES
25  var position = document.createElement("UL");
26  field.appendChild(position);
27
28  // Called on each CP position selector
29  function cpValueOnChange(index, axis, value) {
30    fiber.setControlPoint(index, axis, Number(value));
31    scene.removeCPHighlight();
32    phantom.cpHighlight(guiStatus.editingFiber, index, 'green');
33    document.getElementById('guiFiberLength').innerHTML = roundToPrecision
          (fiber.length);
34    undobutton.disabled = false;
35  }
36
37  var xpos = document.createElement("LI");
38  var xposlabel = document.createElement("LABEL");
39  xposlabel.innerHTML = "x ";
40  xpos.appendChild(xposlabel);
41  var xvalue = document.createElement("INPUT");
42  xvalue.id = 'xvalue';
43  xvalue.style.width = "65px";
44  xvalue.type = "number";
45  xvalue.step = Math.pow(10, -precision);;
46  xvalue.value = cp[0];
47  xvalue.onchange = function() {
48    this.value = roundToPrecision(this.value);
49    cpValueOnChange(index, 'x', this.value);
50  };
51  xpos.appendChild(xvalue);
52  position.appendChild(xpos);
53
54  var ypos = document.createElement("LI");
55  var yposlabel = document.createElement("LABEL");
56  yposlabel.innerHTML = "y ";
57  ypos.appendChild(yposlabel);
58  var yvalue = document.createElement("INPUT");
59  yvalue.id = 'yvalue';
60  yvalue.style.width = "65px";
61  yvalue.type = "number";
```

```
62    yvalue.step = Math.pow(10, -precision);
63    yvalue.value = cp[1];
64    yvalue.onchange = function() {
65      this.value = roundToPrecision(this.value);
66      cpValueOnChange(index, 'y', this.value);
67    };
68    ypos.appendChild(yvalue);
69    position.appendChild(ypos);
70
71    var zpos = document.createElement("LI");
72    var zposlabel = document.createElement("LABEL");
73    zposlabel.innerHTML = "z ";
74    zpos.appendChild(zposlabel);
75    var zvalue = document.createElement("INPUT");
76    zvalue.id = 'zvalue';
77    zvalue.style.width = "65px";
78    zvalue.type = "number";
79    zvalue.step = Math.pow(10, -precision);
80    zvalue.value = cp[2];
81    zvalue.onchange = function() {
82      this.value = roundToPrecision(this.value);
83      cpValueOnChange(index, 'z', this.value);
84    };
85    zpos.appendChild(zvalue);
86    position.appendChild(zpos);
87
88    // Buttons under position selectors
89    var buttons = document.createElement("LI");
90    position.appendChild(buttons);
91    buttons.innerHTML = "   "
92
93    var ddbutton = document.createElement("BUTTON");
94    ddbutton.id = 'ddbutton';
95    if (guiStatus.dragAndDropping) { //Acting only when hovering over CPs
96      ddbutton.className = 'w3-btn w3-yellow w3-hover-khaki w3-border w3-
          ripple w3-small';
97    } else {
98      ddbutton.className = 'w3-btn w3-hover-yellow w3-border w3-border-white
          w3-small w3-ripple';
99    }
100   ddbutton.tile = "Drag and Drop point to edit it";
101   ddbutton.style = 'margin-top: 10px; margin-bottom: 10px';
102   ddbutton.innerHTML = '<i class="icons">&#xE901;</i>';
103   ddbutton.onclick = function() {
104     undobutton.disabled = false;
105     if (!guiStatus.dragAndDropping) {
106       guiStatus.dragAndDropping = true;
107       this.className = 'w3-btn w3-yellow w3-hover-khaki w3-border w3-
            ripple w3-small';
108       xvalue.disabled = true;
109       yvalue.disabled = true;
110       zvalue.disabled = true;
111       dragAndDrop();
112     } else {
```

```
113       guiStatus.dragAndDropping = false;
114       this.className = 'w3-btn w3-hover-yellow w3-border w3-border-white
              w3-small w3-ripple'
115       xvalue.disabled = false;
116       yvalue.disabled = false;
117       zvalue.disabled = false;
118       scene.removeControls();
119     }
120   }
121   buttons.appendChild(ddbutton);
122
123
124   var undobutton = document.createElement("BUTTON");
125   undobutton.id = 'cpUndoButton';
126   undobutton.tile = "Undo (U)";
127   undobutton.className = 'w3-btn w3-hover-blue w3-border w3-border-white
          w3-small'
128   undobutton.style = ddbutton.style;
129   undobutton.innerHTML = '<i class="icons">&#xE900;</i>';
130   // If nothing to undo, button is disabled. If something to, greenpoint
          of editing is shown.
131   if (
132     former[0] == Number(xvalue.value) &&
133     former[1] == Number(yvalue.value) &&
134     former[2] == Number(zvalue.value)
135   ) {
136     undobutton.disabled = true;
137   }
138
139   // guiStatus.former created earlier is recovered when undoing.
140   undobutton.onclick = function() {
141     scene.removeControls();
142
143     xvalue.value = former[0]; xvalue.onchange();
144     yvalue.value = former[1]; yvalue.onchange();
145     zvalue.value = former[2]; zvalue.onchange();
146
147     if (guiStatus.dragAndDropping) {
148       dragAndDrop();
149     } else {
150       this.disabled = true;
151     }
152   }
153   buttons.appendChild(undobutton);
154
155   // ADD+REMOVE
156   buttons.appendChild(document.createElement("BR"));
157
158   var newcpbutton = document.createElement("BUTTON");
159   newcpbutton.style.float = "right";
160   newcpbutton.className = 'w3-btn w3-hover-green w3-border w3-border-white
          w3-small w3-ripple'
161   newcpbutton.innerHTML = "New CP";
162   newcpbutton.onmouseenter = function() { newCPonmouseover(guiStatus.
```

```
        editingFiber, guiStatus.editingCP); };
163   newcpbutton.onmouseleave = function() { fiber = newCPonmouseout(
        guiStatus.editingFiber, guiStatus.editingCP); } // It is necessary
        to renovate the reference
164   newcpbutton.onclick = function() { newCPclick(guiStatus.editingFiber,
        guiStatus.editingCP); }
165
166
167   var removecpbutton = document.createElement("BUTTON");
168   removecpbutton.style.float = "right";
169   removecpbutton.className = 'w3-btn w3-hover-red w3-border w3-border-
        white w3-small w3-ripple'
170   removecpbutton.innerHTML = "Remove CP";
171   removecpbutton.id = 'removecpbutton';
172   removecpbutton.title = "Remove CP (Del)"
173   removecpbutton.onclick = function() { removeCPclick(guiStatus.
        editingFiber, guiStatus.editingCP); }
174
175   // As style is float, must be appended from right to left
176   buttons.appendChild(removecpbutton);
177   buttons.appendChild(newcpbutton);
178
179   if (!((index != 0) & (index + 1 < fiber.controlPoints.length))) {
180     removecpbutton.disabled = true;
181   }
182   if (!(index + 1 < fiber.controlPoints.length)) {
183     newcpbutton.disabled = true;
184   }
185 }
186
187 // Removes the whole CP edit field and creates a new one (except the CP
      edit field). Useful when attempting to refresh.
188 function exitCPedit() {
189 /** @function exitCPedit
190   * @memberof module:GUI Construction
191   * @desc Removes former Control Point edition UI.
192   <br>Restores {@link guiStatus}.
193   */
194   var editGUI = document.getElementById('editGUI');
195   var cpTable = document.getElementById("cpTable");
196
197   editGUI.removeChild(cpTable);
198   scene.removeControls();
199   scene.removeCPHighlight(true);
200   guiStatus.editing('CP', undefined);
201
202   addCPselect();
203   resizeGUI();
204 }
```

2.3.8 fiberedit.js

```
1  /**@overview Contains functions regarding the fiber edition GUI.*/
2
```

```
3  function fiberEdit( index ) {
4  /** @function fiberEdit
5    * @memberof module:GUI Construction
6    * @param {Number} index The index in the array of the fiber to edit.
7    * @desc Adds the fiber edition GUI.
8  */
9    scene.removeCPHighlight(true);
10
11   // editGUI is emptied
12   var editGUI = document.getElementById('editGUI');
13   editGUI.innerHTML = "";
14   editGUI.style = "list-style-type: none";
15
16   // REMOVE BUTTON
17   var removebutton = document.createElement("BUTTON");
18   removebutton.style.float = "right";
19   removebutton.innerHTML = "Remove Fiber";
20   removebutton.id = "removebutton";
21   removebutton.title = "Remove Fiber (Del)"
22   removebutton.className = "w3-btn w3-hover-red w3-border w3-border-white"
23   removebutton.onclick = function() { removeFiberClick() };
24   editGUI.appendChild(removebutton);
25   editGUI.appendChild(document.createElement("BR"));
26   editGUI.appendChild(document.createElement("BR"));
27
28   // PROPERTY FIELD
29   var field = document.createElement("FIELDSET");
30   editGUI.appendChild(field);
31   var fiberprops = document.createElement("UL");
32
33   // NUMBER OF CONTROL POINTS AND COLOR
34   var controlPointsAndColor = document.createElement("LEGEND");
35   var colorSpan = document.createElement("span");
36   colorSpan.style.color = phantom.fibers.tube[index].color.getStyle();
37   colorSpan.style.fontSize = 'x-large';
38   colorSpan.innerHTML = '  &#9632;';
39
40   var controlPointsSpan = document.createElement("span");
41   controlPointsSpan.id = 'guiFiberTitle';
42   controlPointsSpan.innerHTML = phantom.fibers.source[index].controlPoints
         .length + " Points";
43
44   controlPointsAndColor.appendChild(controlPointsSpan);
45   controlPointsAndColor.appendChild(colorSpan);
46
47   field.appendChild(controlPointsAndColor);
48
49   // LENGTH
50   var length = document.createElement("LI");
51   var lengthspan = document.createElement("SPAN");
52   length.appendChild(lengthspan);
53
54   lengthspan.id = "guiFiberLength";
55   lengthspan.innerHTML = Math.floor(phantom.fibers.source[index].length *
```

```
      10) / 10;

  length.innerHTML += " units long";
  fiberprops.appendChild(length);

  // RADIUS
  var radius = document.createElement("LI");
  var radiuslabel = document.createElement("LABEL");
  radiuslabel.innerHTML = "Radius: ";
  var geometry = phantom.fibers.tube[index].mesh.geometry;

  var radiusvalue = document.createElement("INPUT");
  radiusvalue.style.width = "50px";
  radiusvalue.type = "number";
  radiusvalue.min = 0;
  radiusvalue.step = Math.pow(10, -precision);
  radiusvalue.value = phantom.fibers.source[index].radius;
  radiusvalue.onchange = function() {
    this.value = roundToPrecision(Math.max(1 / (10*precision), Math.abs(
        this.value))); //Radius cannot be negative, must be at least
        precision value.
    phantom.fibers.source[index].radius = this.value;
    phantom.fibers.source[index].notify();
  }

  radius.appendChild(radiuslabel);
  radius.appendChild(radiusvalue);
  fiberprops.appendChild(radius);

  // TANGENTS
  var tangentslabel = document.createElement("LABEL");
  tangentslabel.innerHTML = "Tangents: ";

  var tangents = document.createElement("SELECT");
  tangents.style.margin = '3px';
  tangents.onchange = function() {
    phantom.fibers.source[index].tangents = this.value;
    phantom.fibers.source[index].polyCalc();
    phantom.fibers.source[index].notify();
  }
  var symmetric = document.createElement("OPTION");
  symmetric.value = "symmetric";
  symmetric.innerHTML = "Symmetric";
  tangents.options.add(symmetric);
  var incoming = document.createElement("OPTION");
  incoming.value = "incoming";
  incoming.innerHTML = "Incoming";
  tangents.options.add(incoming);
  var outgoing = document.createElement("OPTION");
  outgoing.value = "outgoing";
  outgoing.innerHTML = "Outgoing";
  tangents.options.add(outgoing);

  tangents.value = phantom.fibers.source[index].tangents;
```

```
107   fiberprops.appendChild(tangentslabel);
108   fiberprops.appendChild(tangents);
109   editGUI.appendChild(document.createElement("BR"));
110
111   field.appendChild(fiberprops);
112
113   addCPselect();
114 }
115
116 // This is a separate function so it may be refreshed independently
117 function addCPselect() {
118 /** @function addCPselect
119   * @memberof module:GUI Construction
120   * @desc Adds the control point selector UI for the current fiber.
121   <br>Built in a separate function so it may be refreshed independently.
122   */
123
124   var editGUI = document.getElementById('editGUI');
125
126   // Control Points edition table creation.
127   var table = document.createElement("TABLE");
128   table.id = 'cpTable';
129   // This creates a of the former CP to be used for the Undo Button.
130   phantom.fibers.source[guiStatus.editingFiber].controlPoints.slice(0);
131   editGUI.appendChild(table);
132   // This table contains two cells: left for CP select list and right for
        edit field (when editing)
133   var tr = document.createElement("TR");
134   table.appendChild(tr);
135   var td1 = document.createElement("TD");
136   tr.appendChild(td1);
137   var td2 = document.createElement("TD");
138   tr.appendChild(td2);
139   td2.id = "cpEditor";
140
141   // CONTROL POINTS SELECTION LIST
142   var cplist = document.createElement("UL");
143   cplist.className = 'enabledList';
144   var fiberindex = guiStatus.editingFiber;
145   // cplist.size = phantom.fibers.source[fiberindex].controlPoints.length
        + 1;
146   cplist.id = 'cpSelector';
147   cplist.style.width = '60px'
148   cplist.onmouseenter = function () {
149     if (cplist.childNodes[0].className == 'optionUnselected') {
150       scene.removeCPHighlight();
151       cpEdit(guiStatus.editingCP);
152     } else {
153       scene.removeCPHighlight(true);
154     }
155   };
156   cplist.onmouseleave = function () {
157     guiStatus.retrieve();
158   }
```

```
159
160   // *n* option
161   var option = document.createElement("LI");
162   option.innerHTML = '*n*'
163   option.title = "Exit edit (Esc)"
164   option.className = 'optionSelected';
165   option.onmouseenter = function () {
166     optionOnMouseOver(this);
167   }
168   option.onmouseleave = function() {
169     optionOnMouseLeave(this);
170   }
171   option.onclick = function () {
172     exitCPedit();
173     optionSelect(this);
174   };
175   cplist.appendChild(option);
176
177   // Each CP option
178   phantom.fibers.source[fiberindex].controlPoints.forEach(
179     function(point, index) {
180       var option = document.createElement("LI");
181       option.innerHTML = index.toString();
182       option.className = 'optionUnselected';
183
184       option.onmouseenter = function() {
185         phantom.cpHighlight(fiberindex, index, 'blue');
186         optionOnMouseOver(this);
187       };
188       option.onmouseleave = function() {
189         optionOnMouseLeave(this);
190       };
191       option.onclick = function() {
192         cpSelectClick(fiberindex, index);
193         optionSelect(this);
194       };
195
196       cplist.appendChild(option);
197     }
198   );
199   td1.appendChild(cplist);
200
201   resizeGUI();
202 }
```

2.3.9 handlers.js

```
1  /**@overview Contains handlers for UI elements.*/
2  /**@module GUI Handlers*/
3
4  // SWITCH VIEW BUTTON
5  // Swithes 'preview mode' which will allow the user to display the phantom
       in unfaded mode while editing
6  function switchViewButton() {
```

```
7  /** @function switchViewButton
8    * @memberof module:GUI Handlers
9    * @desc Handler for preview button. Switches fade of the scene.
10 */
11   var button = document.getElementById('switchViewButton');
12
13   if (!guiStatus.previewing) {
14     phantom.addToScene(scene);
15     button.value = "Back";
16     button.className = 'w3-button w3-aqua w3-hover-cyan w3-border w3-block
           w3-ripple';
17     guiStatus.previewing = true;
18   } else {
19     guiStatus.previewing = false;
20     guiStatus.retrieve();
21     button.className = 'w3-btn w3-hover-aqua w3-border w3-block w3-ripple'
         ;
22     button.value = "Preview";
23   }
24
25   if (guiStatus.dragAndDropping) {
26     dragAndDrop();
27   }
28 }
29
30 // disable booleans must be true when the user does not click directly the
       option.
31 // This saves resources by not rebuilding editingGUI and does not reclick
      selectors, which would be annoying.
32 function fiberSelectClick(index, notclicked) {
33 /** @function fiberSelectClick
34   * @memberof module:GUI Handlers
35   * @param {Number} index Index of the fiber in {@link Phantom} array.
36   * @param {Boolean} [notclicked=false] If true, does not change UI or {
         @link guiStatus}
37   object. Useful when changing the scene or previewing.
38   * @desc Events to be fired when a fiber was selected in the list.
39   <br>May be called to tweak the scene.
40   */
41   if (!notclicked) {
42     guiStatus.editing('fiber', index);
43     fiberEdit(index);
44   } else {
45     // We only want CP edition to be undefined when actually clicked!
46     guiStatus.editingFiber = index;
47   }
48   phantom.revealSkeleton(scene, index);
49 }
50 function regionSelectClick(index, notclicked) {
51 /** @function regionSelectClick
52   * @memberof module:GUI Handlers
53   * @param {Number} index Index of the isotropic region in {@link Phantom}
         array.
54   * @param {Boolean} [notclicked=false] If true, does not change UI or {
```

```
      @link guiStatus}
55    object. Useful when changing the scene or previewing.
56    * @desc Events to be fired when an isotropic region was selected in the
          list.
57    <br>May be called to tweak the scene.
58    */
59    guiStatus.editing('region', index);
60    if (!notclicked) {
61      guiStatus.editing('region', index);
62      regionEdit(index);
63    }
64    phantom.addToScene(scene);
65    phantom.regionHighlight(index);
66  }
67  function cpSelectClick(fiberindex, cpindex, notclicked) {
68  /** @function cpSelectClick
69    * @memberof module:GUI Handlers
70    * @param {Number} index Index of the fiber in {@link Phantom} array.
71    * @param {Number} cp Index of the control point in {@link FiberSource}
          array.
72    * @param {Boolean} [notclicked=false] If true, does not change UI or {
          @link guiStatus}
73    object. Useful when changing the scene or previewing.
74    * @desc Events to be fired when a control point was selected in the list
          .
75    <br>May be called to tweak the scene.
76    */
77    if (!notclicked) {
78      guiStatus.formerCP = phantom.fibers.source[fiberindex].controlPoints[
            cpindex].slice(0);
79      guiStatus.dragAndDropping = false;
80      cpEdit(cpindex);
81    }
82    phantom.cpHighlight(fiberindex, cpindex, 'red');
83    phantom.cpHighlight(fiberindex, cpindex, 'green');
84    guiStatus.editing('CP', cpindex);
85    scene.removeControls();
86  }
87
88  // NEW MESH BUTTONS
89  function newFiberClick() {
90  /** @function newFiberClick
91    * @memberof module:GUI Handlers
92    * @desc Fires the creation of a new fiber and goes into edition.
93  */
94    phantom.newFiber();
95    phantom.addToScene(scene);
96
97    setupGUI();
98    selectOption(document.getElementById("fiberSelector"), phantom.fibers.
          source.length);
99    fiberSelectClick(phantom.fibers.source.length - 1);
100 }
101
```

```
102 function newIsotropicRegionClick() {
103 /** @function newIsotropicRegionClick
104   * @memberof module:GUI Handlers
105   * @desc Fires the creation of a new isotropic region and goes into
          edition.
106 */
107
108   phantom.newIsotropicRegion();
109   phantom.addToScene(scene);
110
111   setupGUI();
112   selectOption(document.getElementById("regionSelector"), phantom.
          isotropicRegions.source.length);
113   regionSelectClick(phantom.isotropicRegions.source.length - 1);
114 }
115
116 // REMOVE MESH
117 function removeFiberClick() {
118 /** @function removeFiberClick
119   * @memberof module:GUI Handlers
120   * @desc Fires the removal of a fiber and quits edition. Prompts the user
           for confirmation.
121 */
122   if (window.confirm("Are you sure you want to remove this fiber? This
          action cannot be undone.")) {
123     var index = guiStatus.editingFiber;
124     phantom.fibers.source.splice(index, 1);
125     phantom.fibers.tube.splice(index, 1);
126     phantom.fibers.skeleton.splice(index, 1);
127
128     phantom.addToScene(scene);
129     setupGUI();
130   }
131 }
132 function removeIsotropicRegionClick() {
133 /** @function removeIsotropicRegionClick
134   * @memberof module:GUI Handlers
135   * @desc Fires the removal of an isotropic region and quits edition.
          Prompts the user for confirmation.
136 */
137   if (window.confirm("Are you sure you want to remove this isotropic
          region? This action cannot be undone.")) {
138     var index = guiStatus.editingRegion;
139     phantom.isotropicRegions.source.splice(index, 1);
140     phantom.isotropicRegions.sphere.splice(index, 1);
141
142     phantom.addToScene(scene);
143     setupGUI();
144   }
145 }
146
147 // CP ADD+REMOVE
148 function newCPclick(fiber, cp) {
149 /** @function newCPclick
```

```
150   * @memberof module:GUI Handlers
151   * @param {Number} index Index of the fiber in {@link Phantom} array.
152   * @param {Number} index Index of the control point in {@link FiberSource
          } array.
153   * @desc Fires the addition of a new Control Point after the current one.
           Gets into edit.
154 */
155   // Control point was yet created by hover function; it just needs to be
          formerly added.
156   phantom.addToScene(scene);
157   guiStatus.editing('CP', cp + 1);
158   guiStatus.retrieve();
159   exitCPedit();
160   cpSelectClick(fiber, cp + 1);
161   selectOption(document.getElementById("cpSelector"), cp + 2);
162   document.getElementById("guiFiberTitle").innerHTML = phantom.fibers.
          source[guiStatus.editingFiber].controlPoints.length + " Points";
163   document.getElementById('fiberSelector').childNodes[guiStatus.
          editingFiber + 1].childNodes[1].innerHTML = phantom.fibers.source[
          guiStatus.editingFiber].controlPoints.length + " points";
164 }
165 function newCPonmouseover(fiber, cp) {
166 /** @function newCPonmouseover
167   * @memberof module:GUI Handlers
168   * @param {Number} index Index of the fiber in {@link Phantom} array.
169   * @param {Number} cp Index of the control point in {@link FiberSource}
          array.
170   * @desc Hover for new control point button. Simulates in the scene the
          addition of a new control point in green color.
171 */
172   phantom.addCP(fiber, cp);
173   phantom.addToScene(scene);
174   fiberSelectClick(fiber, true);
175   phantom.cpHighlight(fiber, cp, 'red');
176   phantom.cpHighlight(fiber, cp + 1, 'green');
177   document.getElementById('guiFiberLength').innerHTML = roundToPrecision(
          phantom.fibers.source[guiStatus.editingFiber].length);
178 }
179 function newCPonmouseout(fiber, cp) {
180 /** @function newCPonmouseout
181   * @memberof module:GUI Handlers
182   * @param {Number} index Index of the fiber in {@link Phantom} array.
183   * @param {Number} cp Index of the control point in {@link FiberSource}
          array.
184   * @desc Restores the scene after unhover in new control point button.
185   * @returns {FiberSource} Actual source object.
186 */
187   phantom.removeCP(fiber, cp + 1);
188   phantom.addToScene(scene);
189   guiStatus.retrieve();
190   var source = phantom.fibers.source[guiStatus.editingFiber];
191   document.getElementById('guiFiberLength').innerHTML = roundToPrecision(
          source.length);
192
```

```
193   return source;
194 }
195
196 function removeCPclick(fiber, cp) {
197 /** @function removeCPclick
198   * @memberof module:GUI Handlers
199   * @desc Fires the removal of a control point and quits edition. Prompts
          the user for confirmation.
200 */
201   if (window.confirm("Are you sure you want to remove this control point?
          This action cannot be undone.")) {
202     phantom.removeCP(fiber, cp);
203     phantom.addToScene(scene);
204     guiStatus.editing('CP', undefined);
205     guiStatus.retrieve();
206     exitCPedit();
207     document.getElementById('guiFiberLength').innerHTML = roundToPrecision
          (phantom.fibers.source[guiStatus.editingFiber].length);
208     document.getElementById("guiFiberTitle").innerHTML = phantom.fibers.
          source[guiStatus.editingFiber].controlPoints.length + " Points";
209     document.getElementById('fiberSelector').childNodes[guiStatus.
          editingFiber + 1].childNodes[1].innerHTML = phantom.fibers.source[
          guiStatus.editingFiber].controlPoints.length + " points";
210   }
211 }
212
213 // AXES TOGGLE
214 function toggleAxes() {
215 /** @function toggleAxes
216   * @memberof module:GUI Handlers
217   * @desc Toogle axes view button. Switches between showing or removing in
          the scene.
218 */
219   button = document.getElementById('toggleAxesButton');
220   name = 'axes';
221   var length = phantom.radius() * 1.5;
222
223   if (scene.getObjectByName(name)) {
224     scene.remove(scene.getObjectByName(name))
225     button.className = 'w3-btn w3-border w3-hover-deep-purple w3-block w3-
          ripple';
226   } else {
227     var axes = buildAxes(length);
228     axes.name = name;
229     scene.add(axes);
230     button.className = 'w3-button w3-deep-purple w3-hover-indigo w3-border
           w3-block w3-ripple';
231   }
232   render();
233 }
234 // PLANE SELECTORS
235 // Double click for inverted axis was commented for it to be disabled for
      the moment. Found it annoying when attempting to move points.
236 function moveCameraXY() {
```

```
237 /** @function moveCameraXY
238   * @memberof module:GUI Handlers
239   * @desc Moves view to the XY plane.
240 */
241   camera.up = new THREE.Vector3(0, 1, 0);
242   controls.target = new THREE.Vector3(0, 0, 0);
243   // if (camera.position.z == phantom.radius()*1.5) {
244     camera.position.set(0, 0, 0);
245     camera.position.z = phantom.radius() * 2 * 1.5;
246   // }
247   // else {
248   //   camera.position.set(0, 0, 0);
249   //   camera.position.z = phantom.radius()*1.5;
250   // }
251 }
252 function moveCameraXZ() {
253 /** @function moveCameraXZ
254   * @memberof module:GUI Handlers
255   * @desc Moves view to the XZ plane.
256 */
257   camera.up = new THREE.Vector3(0, 0, 1);
258   controls.target = new THREE.Vector3(0, 0, 0);
259   // if (camera.position.y == phantom.radius()*1.5) {
260     camera.position.set(0, 0, 0);
261     camera.position.y = phantom.radius() * 2 * -1.5;
262   // }
263   // else {
264   //   camera.position.set(0, 0, 0);
265   //   camera.position.y = phantom.radius()*-1.5;
266   // }
267 }
268 function moveCameraZY() {
269 /** @function moveCameraZY
270   * @memberof module:GUI Handlers
271   * @desc Moves view to the ZY plane.
272 */
273   camera.up = new THREE.Vector3(0, 1, 0);
274   controls.target = new THREE.Vector3(0, 0, 0);
275   // if (camera.position.x == phantom.radius()*1.5) {
276     camera.position.set(0, 0, 0);
277     camera.position.x = phantom.radius() * 2 * -1.5;
278   // }
279   // else {
280   //   camera.position.set(0, 0, 0);
281   //   camera.position.x = phantom.radius()*-1.5;
282   // }
283 }
284
285
286 // OPACITY
287 function opacitySelectChange(selector) {
288 /** @function opacitySelectChange
289   * @memberof module:GUI Handlers
290   * @param {DOM} selector Opacity selector DOM element.
```

```
291   * @desc Fired when value in the opacity selector is changed. Corrects
          the value and fires the scene change.
292 */
293   // Make the value stay between min and max.
294   if (Number(selector.value) > Number(selector.max)) {
295     selector.value = selector.max;
296   } else if (Number(selector.value) < Number(selector.min)) {
297     selector.value = selector.min;
298   }
299   // Allow custom step; do not allow decimal values.
300   selector.value = Math.round(Number(selector.value));
301
302   phantom.highlightOpacity = selector.value / 100;
303   guiStatus.retrieve();
304 }
305
306 function saveClick() {
307 /** @function saveClick
308   * @memberof module:GUI Handlers
309   * @desc Pushes the download of the current Phantom.
310 */
311   pushDownload(phantom.export());
312 }
```

2.3.10 regionedit.js

```
1  /**@overview Contains functions regarding the isotropic region edition GUI
       .*/
2
3  function regionEdit(index) {
4  /** @function regionEdit
5    * @memberof module:GUI Construction
6    * @param {Number} index The index in the array of the fiber to edit.
7    * @desc Adds the isotropic region edition GUI.
8    */
9
10   resizeGUI();
11   scene.removeCPHighlight(true);
12
13   // editGUI is emptied
14   var editGUI = document.getElementById('editGUI');
15   editGUI.innerHTML = "";
16   editGUI.style = "list-style-type: none";
17
18   // REMOVE BUTTON
19   var removebutton = document.createElement("BUTTON");
20   removebutton.style.float = "right";
21   removebutton.innerHTML = "Remove Isotropic Region";
22   removebutton.id = "removebutton";
23   removebutton.id = "Remove Isotropic Region (Del)"
24   removebutton.className = "w3-btn w3-hover-red w3-border w3-border-white"
25   removebutton.onclick = function() { removeIsotropicRegionClick() };
26   editGUI.appendChild(removebutton);
27   editGUI.appendChild(document.createElement("BR"));
```

```
28  editGUI.appendChild(document.createElement("BR"));
29
30  // PROPERTY FIELD
31  var field = document.createElement("FIELDSET");
32  editGUI.appendChild(field);
33  var regionprops = document.createElement("UL");
34
35  // TITLE AND COLOR
36  var titleAndColor = document.createElement("LEGEND");
37  var colorSpan = document.createElement("span");
38  colorSpan.style.color = phantom.isotropicRegions.sphere[index].color.
        getStyle();
39  colorSpan.style.fontSize = 'x-large';
40  colorSpan.innerHTML = '  &#9632;';
41
42  var titleSpan = document.createElement("span");
43  titleSpan.innerHTML = "Editing region ";
44
45  titleAndColor.appendChild(titleSpan);
46  titleAndColor.appendChild(colorSpan);
47
48  field.appendChild(titleAndColor);
49
50  // RADIUS
51  var radius = document.createElement("LI");
52  var radiuslabel = document.createElement("LABEL");
53  radiuslabel.innerHTML = "Radius: ";
54  var geometry = phantom.isotropicRegions.sphere[index].mesh.geometry;
55
56  var radiusvalue = document.createElement("INPUT");
57  radiusvalue.style.width = "50px";
58  radiusvalue.type = "number";
59  radiusvalue.min = 0;
60  radiusvalue.step = Math.pow(10, -precision);
61  radiusvalue.value = phantom.isotropicRegions.source[index].radius;
62  radiusvalue.onchange = function() {
63    this.value = roundToPrecision(Math.max(1 / (10*precision), Math.abs(
          this.value))); //Radius cannot be negative, must be at least
          precision value.
64    phantom.isotropicRegions.source[index].setRadius(this.value);
65    // Update the radius value in the region selector list
66    document.getElementById('regionSelector').childNodes[index + 1].
          childNodes[1].innerHTML = 'radius ' + this.value;
67  }
68
69  radius.appendChild(radiuslabel);
70  radius.appendChild(radiusvalue);
71  regionprops.appendChild(radius);
72
73  // POSITION
74  var position = document.createElement("FIELDSET");
75  position.innerHTML = "<legend>Position</legend>";
76  var positionul = document.createElement("UL");
77  position.appendChild(positionul);
```

```
78
79  var xpos = document.createElement("LI");
80  var xposlabel = document.createElement("LABEL");
81  xposlabel.innerHTML = "x ";
82  xpos.appendChild(xposlabel);
83  var xvalue = document.createElement("INPUT");
84  xvalue.id = 'xvalue';
85  xvalue.style.width = "60px";
86  xvalue.type = "number";
87  xvalue.step = Math.pow(10, -precision);
88  xvalue.value = phantom.isotropicRegions.source[index].center[0];
89  xvalue.onchange = function() {
90    this.value = roundToPrecision(this.value);
91    phantom.isotropicRegions.source[index].setCenter('x', xvalue.value);
92  }
93  xpos.appendChild(xvalue);
94  positionul.appendChild(xpos);
95
96  var ypos = document.createElement("LI");
97  var yposlabel = document.createElement("LABEL");
98  yposlabel.innerHTML = "y ";
99  ypos.appendChild(yposlabel);
100 var yvalue = document.createElement("INPUT");
101 yvalue.id = 'yvalue';
102 yvalue.style.width = "60px";
103 yvalue.type = "number";
104 yvalue.step = Math.pow(10, -precision);
105 yvalue.value = phantom.isotropicRegions.source[index].center[1];
106 yvalue.onchange = function() {
107   this.value = roundToPrecision(this.value);
108   phantom.isotropicRegions.source[index].setCenter('y', yvalue.value);
109 }
110 ypos.appendChild(yvalue);
111 positionul.appendChild(ypos);
112
113 var zpos = document.createElement("LI");
114 var zposlabel = document.createElement("LABEL");
115 zposlabel.innerHTML = "z ";
116 zpos.appendChild(zposlabel);
117 var zvalue = document.createElement("INPUT");
118 zvalue.id = 'zvalue';
119 zvalue.style.width = "60px";
120 zvalue.type = "number";
121 zvalue.step = Math.pow(10, -precision);
122 zvalue.value = phantom.isotropicRegions.source[index].center[2];
123 zvalue.onchange = function() {
124   this.value = roundToPrecision(this.value);
125   phantom.isotropicRegions.source[index].setCenter('z', zvalue.value);
126 }
127 zpos.appendChild(zvalue);
128 positionul.appendChild(zpos);
129
130 regionprops.appendChild(position);
131
```

```
132   field.appendChild(regionprops);
133 }
```

2.3.11 resize.js

```
1  /**@overview Contains resizeGUI() function*/
2  /**@module GUI Managers*/
3
4  // Resizes selector objects so those fit in the screen
5  function resizeGUI() {
6  /** @function resizeGUI
7    * @memberof module:GUI Managers
8    * @desc Resizes element selector lists so those just take the amount of
         space
9    left in the screen. This avoids overflows and maintains the page size
         same as
10   the window size.
11   <br>Called when elements are added or removed and when window is resized
         .
12   */
13
14   // PRIVATE FUNCTION Returns amout of window height in text lines
15   function countDocumentLines() {
16     // From http://stackoverflow.com/questions/4392868/javascript-find-
           divs-line-height-not-css-property-but-actual-line-height
17     function getLineHeight(element){
18       var temp = document.createElement(element.nodeName);
19       temp.setAttribute("style","margin:0px;padding:0px;font-family:"+
             element.style.fontFamily+";font-size:"+element.style.fontSize);
20       temp.innerHTML = "test";
21       temp = element.parentNode.appendChild(temp);
22       var ret = temp.clientHeight;
23       temp.parentNode.removeChild(temp);
24       return ret;
25     }
26     var divHeight = window.innerHeight;
27     var lineHeight = getLineHeight(document.getElementById('leftGUI'));
28     var lines = Math.floor(divHeight / lineHeight);
29     return lines;
30   }
31
32   // Lines is the height amount in lines left for the gui elements.
33   var lines = countDocumentLines() - 9;
34   var leftGUI = document.getElementById("leftGUI");
35   var fiberSelector = document.getElementById("fiberSelector");
36   var regionSelector = document.getElementById("regionSelector");
37
38   // Width is subtracted 10 pixels for allowing space to scrollbar.
39   fiberSelector.style.width = (leftGUI.offsetWidth - 15).toString() + 'px'
         ;
40   regionSelector.style.width = (leftGUI.offsetWidth - 15).toString() + 'px
         ';
41
42   // The resizable elements are selectors. We subtract space taken by
```

```
      other gui elements if those are present.
43    if (guiStatus.editingFiber !== undefined) {
44      lines -= 12 + Math.max(phantom.fibers.source[guiStatus.editingFiber].
            controlPoints.length + 1, 12);
45      var cpEditor = document.getElementById("cpEditor");
46      cpEditor.style.width = (leftGUI.offsetWidth - 65).toString() + 'px';
47    } else if (guiStatus.editingRegion !== undefined) {
48      lines -= 16;
49    }
50
51    // +1 is due to *none* option.
52    var fiberNumber = (phantom.fibers.source.length + 1)
53    var regionNumber = (phantom.isotropicRegions.source.length + 1)
54
55    var minsize = 3;
56
57    // Final size to be between total number of elements (no select scroll)
          and minsize
58    // 1.2 factor is due to line height correction
59    fiberSelector.style.height = (Math.min( Math.max(lines - regionNumber,
          minsize), fiberNumber) * 1.2).toString() + 'em';
60    regionSelector.style.height = (Math.min( Math.max(lines - fiberNumber,
          minsize), regionNumber) * 1.2).toString() + 'em';
61  }
```

2.3.12 setup.js

```
1   /** @overview Contains basic GUI constructors.*/
2   /** @module GUI Construction */
3   /** @var {guiStatus} guiStatus
4    * @desc Global variable indicating the current task the GUI is performing
         .
5    */
6   var guiStatus;
7
8   function setupGUI() {
9     /** @function setupGUI
10      * @memberof module:GUI Construction
11      * @desc Constructs basic-static GUI when no action has taken place yet
            .
12      Defines {@link guiStatus} global variable.
13      <br>Adds event listeners to window object for keyboard bindings.
14      */
15    guiStatus = new GuiStatus();
16    resizeGUI();
17
18    var fiberSelector = document.getElementById("fiberSelector");
19    var regionSelector = document.getElementById("regionSelector");
20    // Empty them - this function might be called for GUIupdating
21    fiberSelector.innerHTML = "";
22    regionSelector.innerHTML = "";
23
24    if (phantom.fibers.source.length > 0) {
25      // Add *none* option
```

```
26      var option = document.createElement("LI");
27      option.innerHTML = '*none*'
28      option.title = "Exit edit (Esc)"
29      option.className = 'optionSelected';
30      // If any fiber is being edited, move to non-edit mode
31      option.onclick = function() {
32        if (guiStatus.editingRegion === undefined) {
33          guiStatus.unediting();
34          guiStatus.retrieve();
35          optionSelect(this);
36        };
37        resizeGUI();
38      };
39      option.onmouseenter = function() {
40        phantom.addToScene(scene);
41        optionOnMouseOver(this);
42      };
43      option.onmouseleave = function() {
44        optionOnMouseLeave(this);
45      };
46      fiberSelector.appendChild(option);
47      fiberSelector.className = 'enabledList';
48
49      // Add the rest of the options
50      phantom.fibers.source.forEach(function(fiber, index) {
51        var backgroundColor = fiber.color;
52
53        var option = document.createElement("LI");
54        var selectColorSpan = document.createElement("span");
55        var selectTextSpan = document.createElement("span");
56        selectColorSpan.style.color = backgroundColor.getStyle();
57        selectColorSpan.innerHTML = '&#9632; ';
58
59        selectTextSpan.innerHTML = fiber.controlPoints.length.toString() + "
              points";
60
61        option.appendChild(selectColorSpan);
62        option.appendChild(selectTextSpan);
63        option.className = 'optionUnselected';
64
65        option.onmouseenter = function() {
66          phantom.fiberHighlight(index);
67          optionOnMouseOver(this);
68        };
69        option.onmouseleave = function() {
70          optionOnMouseLeave(this);
71        };
72        option.onclick = function() {
73          fiberSelectClick(index);
74          optionSelect(this)
75        };
76        fiberSelector.appendChild(option);
77      });
78    } else {
```

```
79      var option = document.createElement("LI");
80      option.innerHTML = '(any)';
81      fiberSelector.appendChild(option);
82      fiberSelector.className = 'disabledList';
83    }
84
85    if (phantom.isotropicRegions.source.length > 0) {
86      // Add *none* option
87      var option = document.createElement("LI");
88      option.innerHTML = '*none*'
89      option.title = "Exit edit (Esc)"
90      option.className = 'optionSelected';
91      // If any fiber is being edited, move to non-edit mode
92      option.onclick = function() {
93        if (guiStatus.editingFiber === undefined) {
94          guiStatus.unediting();
95          guiStatus.retrieve();
96          optionSelect(this);
97        };
98        resizeGUI();
99      };
100     option.onmouseenter = function() {
101       phantom.addToScene(scene);
102       optionOnMouseOver(this);
103     };
104     option.onmouseleave = function() {
105       optionOnMouseLeave(this);
106     };
107     regionSelector.appendChild(option);
108     regionSelector.className = 'enabledList';
109
110     // Add the rest of the options
111     phantom.isotropicRegions.source.forEach(function(region, index) {
112       var backgroundColor = region.color;
113
114       var option = document.createElement("LI");
115       option.className = 'optionUnselected';
116       var selectColorSpan = document.createElement("span");
117       var selectTextSpan = document.createElement("span");
118       selectColorSpan.style.color = backgroundColor.getStyle();
119       selectColorSpan.innerHTML = '&#9632; ';
120
121       selectTextSpan.innerHTML = "radius " + region.radius.toString();
122
123       option.appendChild(selectColorSpan);
124       option.appendChild(selectTextSpan);
125
126       option.onmouseover = function() {
127         phantom.regionHighlight(index);
128         optionOnMouseOver(this);
129       };
130       option.onmouseleave = function() {
131         optionOnMouseLeave(this);
132       };
```

```
133       option.onclick = function() {
134         regionSelectClick(index);
135         optionSelect(this)
136       };
137       regionSelector.appendChild(option);
138     });
139   } else {
140     // var option = document.createElement("LI");
141     regionSelector.innerHTML = '(any)'
142     // regionSelector.appendChild(option);
143     regionSelector.className = 'disabledList';
144   }
145
146   // Add keyboard shortcuts
147   window.addEventListener('keyup', function(e) {
148     if (document.hasFocus()) { //Prevents events from firing when not
          being focused on the app
149       switch (e.keyCode) {
150         case 27: //Esc
151           if (guiStatus.editingFiber + 1) {
152             if (guiStatus.editingCP + 1) {
153               document.getElementById('cpSelector').childNodes[0].onclick
                    ();
154               optionOnMouseLeave(document.getElementById('cpSelector').
                    childNodes[0]);
155             } else {
156               fiberSelector.childNodes[0].onclick();
157               optionOnMouseLeave(fiberSelector.childNodes[0]);
158             }
159           }
160           if (guiStatus.editingRegion + 1) {
161             regionSelector.childNodes[0].onclick();
162             optionOnMouseLeave(regionSelector.childNodes[0]);
163           }
164           break;
165         case 80: //P
166           if (guiStatus.editingFiber | guiStatus.editingRegion) {
167             switchViewButton();
168           }
169           break;
170         case 65: //A
171           toggleAxes();
172           break;
173         case 88: //X
174           moveCameraZY();
175           break;
176         case 89: //Y
177           moveCameraXZ();
178           break;
179         case 90: //Z
180           moveCameraXY();
181           break;
182         case 83: //S
183           saveClick();
```

```
184           break;
185         case 68: //D
186           if (document.getElementById('ddbutton')) { //If does not exist
                , won't fire.
187             document.getElementById('ddbutton').onclick();
188           }
189           break;
190         case 46: //Del
191           if (guiStatus.editingFiber + 1) {
192             if (guiStatus.editingCP + 1) {
193               if (document.getElementById('removecpbutton')) { //If does
                    not exist, won't fire.
194                 document.getElementById('removecpbutton').onclick();
195               }
196             } else {
197               removeFiberClick();
198             }
199           } else if (guiStatus.editingRegion + 1) {
200             removeIsotropicRegionClick();
201           }
202           break;
203         case 85: //U
204           if (document.getElementById('cpUndoButton')) {
205             if (!document.getElementById('cpUndoButton').disabled) {
206               document.getElementById('cpUndoButton').click();
207             }
208           }
209       }
210     }
211   });
212
213   document.getElementById('opacitySelector').value = phantom.
        highlightOpacity * 100;
214   editExit();
215 }
216
217 // Add element buttons are only available when no element is being edited
218 function editExit() {
219   /** @function editExit
220    * @memberof module:GUI Construction
221    * @desc Removes any edition UI. Adds new element buttons.
222    */
223   scene.remove(dragAndDrop); //In case of being present
224   var editGUI = document.getElementById('editGUI');
225   editGUI.innerHTML = ""
226
227
228   var newfiberbutton = document.createElement("BUTTON");
229   newfiberbutton.style.float = "right";
230   newfiberbutton.innerHTML = "New Fiber";
231   newfiberbutton.className = "w3-btn w3-hover-green w3-border w3-border-
        white"
232   newfiberbutton.onclick = function() {
233     newFiberClick()
```

```
234   };
235
236   var newregionbutton = document.createElement("BUTTON");
237   newregionbutton.style.float = "right";
238   newregionbutton.innerHTML = "New Region";
239   newregionbutton.className = "w3-btn w3-hover-green w3-border w3-border-
          white"
240   newregionbutton.onclick = function() {
241     newIsotropicRegionClick()
242   };
243
244   // As style is float, must be appended from right to left
245   editGUI.appendChild(newregionbutton);
246   editGUI.appendChild(newfiberbutton);
247 }
```

2.3.13 status.js

```
1  /**@overview Contains class GuiStatus and its modules.*/
2
3  function GuiStatus() {
4  /** @class GuiStatus
5    * @memberof module:GUI Managers
6    * @classdesc Class used for defining current app GUI status.
7    * @prop {Number} editingFiber=undefined; Index of currently being edited
           fiber. If any, undefined.
8    * @prop {Number} editingCP=undefined; Index of currently being edited
         control point. If any, undefined.
9    * @prop {Number} editingRegion=undefined; Index of currently being
         edited isotropic region. If any, undefined.
10   * @prop {Boolean} previewing=false Whether preview mode is active or not
         .
11   * @prop {Boolean} dragAndDropping=false Whether drag and drop control
         point edit mode is active or not.
12   */
13
14     this.previewing = false;
15     document.getElementById("switchViewButton").disabled = true;
16
17     this.dragAndDropping = false;
18
19     this.editingFiber = undefined;
20     this.editingCP = undefined;
21     this.editingRegion = undefined;
22 }
23
24 GuiStatus.prototype = {
25   editing: function(element, index) {
26   /** @function editing
27     * @memberof module:GUI Managers.GuiStatus
28     * @param {String} element Element to be edited. 'fiber', 'CP' or '
           region'.
29     * @param {Number} index Index of the element in its array.
30     * @desc Changes the properties of the object matching the specified
```

```
        input.
31      */
32      switch (element) {
33        case 'fiber':
34          this.unediting();
35          this.editingFiber = index;
36          break;
37        case 'CP':
38          if (this.editingFiber === undefined) {
39            console.error('Tried to edit CP with any fiber in edit!');
40            break;
41          }
42          this.editingCP = index;
43          break;
44        case 'region':
45          this.unediting();
46          this.editingRegion = index;
47          break;
48        default: console.error('Element string in status was not correct');
49      }
50
51      document.getElementById("switchViewButton").disabled = false;
52      if (!this.previewing) {
53        document.getElementById("switchViewButton").className = 'w3-btn w3-
            border w3-hover-aqua w3-block w3-ripple';
54      }
55    },
56    retrieve: function() {
57    /** @function retrieve
58      * @memberof module:GUI Managers.GuiStatus
59      * @desc Turns the scene into the current status. Refreshes the GUI.
60      */
61      if (this.previewing) {
62        phantom.addToScene(scene);
63      } else {
64        if (this.editingFiber !== undefined) {
65          fiberSelectClick(this.editingFiber, true);
66          if (this.editingCP !== undefined) {
67            cpSelectClick(this.editingFiber, this.editingCP, true);
68            if (guiStatus.dragAndDropping) {
69              guiStatus.dragAndDropping = false; //Simulate D&D bare click
70              document.getElementById('ddbutton').onclick();
71            }
72          }
73        } else if (this.editingRegion !== undefined) {
74          regionSelectClick(this.editingRegion, true)
75        } else {
76          phantom.addToScene(scene);
77          editExit();
78        }
79      }
80    },
81    unediting: function() {
82    /** @function unediting
```

```
83      * @memberof module:GUI Managers.GuiStatus
84      * @desc Turns the scene into unediting status. Restores the GUI.
85      */
86      this.previewing = false;
87      this.dragAndDropping = false;
88      scene.removeControls();
89
90      document.getElementById("switchViewButton").value = "Preview";
91      document.getElementById("switchViewButton").disabled = true;
92      document.getElementById("switchViewButton").className = 'w3-btn w3-
            border w3-hover-aqua w3-block w3-ripple';
93
94      this.editingFiber = undefined;
95      this.editingCP = undefined;
96      this.editingRegion = undefined;
97    }
98  }
```

2.3.14 stylehandlers.js

```
1   /**@overview Contains all functions regarding selection lists hover*/
2
3   function optionOnMouseOver(option) {
4   /** @function optionOnMouseOver
5     * @memberof module:GUI Handlers
6     * @param {DOM} option Parent option DOM element.
7     * @desc Applies hover classes to onmouseovered options in lists.
8     */
9     if (option.className == 'optionSelected') {
10      option.className = 'optionSelectedAndOnMouseOver';
11    } else if (option.className == 'optionUnselected') {
12      option.className = 'optionOnMouseOver';
13    }
14  }
15  function optionOnMouseLeave(option) {
16  /** @function optionOnMouseLeave
17    * @memberof module:GUI Handlers
18    * @param {DOM} option Parent option DOM element.
19    * @desc Restores class to onmouseovered options in lists.
20    */
21    if (option.className == 'optionSelectedAndOnMouseOver') {
22      option.className = 'optionSelected';
23    } else if (option.className == 'optionOnMouseOver') {
24      option.className = 'optionUnselected';
25    }
26  }
27
28  function optionSelect(option) {
29  /** @function optionSelect
30    * @memberof module:GUI Handlers
31    * @param {DOM} option Parent option DOM element.
32    * @desc Applies selected class to selected options in lists. Removes
          selected options from other lists.
33    */
```

```
34    var fiberSelector = document.getElementById("fiberSelector");
35    var regionSelector = document.getElementById("regionSelector");
36    if ((option.parentNode == fiberSelector) | (option.parentNode ==
          regionSelector)) {
37      var fiberoptions = Array.from(fiberSelector.childNodes);
38      var regionoptions = Array.from(regionSelector.childNodes);
39
40      fiberoptions.forEach( function( element ) {
41        element.className = 'optionUnselected';
42      });
43      regionoptions.forEach( function( element ) {
44        element.className = 'optionUnselected';
45      });
46
47      if (option.parentNode == fiberSelector) {
48        if (regionoptions.length > 1) {  //Avoids selecting (any) option.
49          regionoptions[0].className = 'optionSelected';
50        }
51      } else {
52        if (fiberoptions.length > 1) {  //Avoids selecting (any) option.
53          fiberoptions[0].className = 'optionSelected';
54        }
55      }
56    } else {
57      var listoptions = Array.from(option.parentNode.childNodes);
58      listoptions.forEach( function( element ) {
59        element.className = 'optionUnselected';
60      });
61    }
62    option.className = 'optionSelectedAndOnMouseOver'
63  }
64
65  function selectOption(list, number) {
66  /** @function selectOption
67    * @memberof module:GUI Handlers
68    * @param {DOM} list List's DOM element.
69    * @param {Number} number Index of option to select
70    * @desc Changes class of option as if it was clicked, given the list and
          its index.
71    */
72    var options = Array.from(list.childNodes);
73    optionSelect(options[number]);
74    optionOnMouseLeave(options[number]);
75  }
```

2.3.15 FiberSource.js

```
1   /** @overview Contains Class definitions for {@link FiberSource} and {
        @link IsotropicRegionSource}. */
2
3   /** @constant colors
4     * @desc An HEX array used as a color library for random generation.
5   */
6   var colors = [0xFF1E00, 0xFFB300, 0x1533AD, 0x00BF32, 0xBF4030,
```

```
7           0xBF9430, 0x2C3D82, 0x248F40, 0xA61300, 0xA67400,
8           0x071C71, 0x007C21, 0xFF5640, 0xFFC640, 0x4965D6,
9           0x38DF64, 0xFF8373, 0xFFD573, 0x6F83D6, 0x64DF85,
10          0xFF5600, 0xFF7C00, 0x04859D, 0x00AA72, 0x60D4AE,
11          0xBF6030, 0xBF7630, 0x206876, 0x207F60, 0x5FBDCE,
12          0xA63800, 0xA65100, 0x015666, 0x006E4A, 0xFFB773,
13          0xFF8040, 0xFF9D40, 0x37B6CE, 0x35D4A0, 0xFFA273];
14
15
16  function FiberSource(controlPoints, tangents, radius, color) {
17    /** @class FiberSource
18    * @classdesc A fiber bundle in Phantomas is defined as a cylindrical
          tube wrapped around
19    its centerline. The centerline itself is a continuous curve in 3D, and
          can be
20    simply created from a few control points. All the fibers created are
          supposed to connect two
21    cortical areas.<br>
22    * FiberSource is the basic Class for the representation of a Fiber.
          Objects containing
23    the geometries to be added to the scene are to be referred to
          FiberSource for
24    gathering any necessary information.
25
26    * @param {array} controlPoints Array-of-arrays (N, 3) containing the 3-D
           coordinates
27    of the fiber Control Points.
28    * @param {string} [tangents='symmetric'] Way the tangents are to be
          computed.
29    Available options: 'incoming', 'outgoing', 'symmetric'
30    * @param {number} [radius=1] Fiber radius; same dimensions as
          controlPoints.
31    * @param {number} [color] Color in which the fiber should be displayed.
          If not
32    specified, to be picked randomly from {@link colors}.
33
34    * @property {array} observers Objects to be notified when some change is
           applied
35    */
36
37    // Initialize properties. By default tangents = 'symmetric', radius = 1.
38    this.controlPoints = controlPoints;
39
40    if (tangents === undefined) {
41      tangents = 'symmetric';
42    }
43    this.tangents = tangents;
44
45    if (radius === undefined) {
46      radius = 1;
47    }
48    this.radius = radius;
49
50    if (color === undefined) {
```

```
51      color = new THREE.Color(colors[Math.floor(Math.random()*colors.length)
            ]);
52    }
53    this.color = new THREE.Color(color);
54
55    // Calculate coefficients
56    this.polyCalc();
57
58    // FiberSource objects will act as subjects to FiberTube or
          FiberSkeleton
59    this.observers = [];
60  }
61
62  FiberSource.prototype = {
63    polyCalc: function() {
64    /** @function polyCalc
65      * @memberof FiberSource
66      * @desc When called, coefficients are calculated.
67        This takes the FiberSource instance from control points, and a
              specified
68        mode to compute the tangents.<br>
69        The output is the coefficients as <p style='margin-left: 10em; font-
              family:monospace'>
70        (x,y,z)(t)= a + b[(t-ti)/(ti1-ti)] + c[(t-ti)/(ti1-ti)]^2 + d[(t-ti)
              /(ti1-ti)]^3</p> for each x, y and
71        z and for each pair of points, as <i>this.xpoly</i>, <i>this.ypoly</
              i> and <i>this.zpoly</i>.
72        Timestamps normalized in <[0,1] are also calculated in <i>this.ts</i
              >.
73     */
74     // Take distance of each pair of given control points
75      nbPoints = this.controlPoints.length;
76      var distances = [];
77      for (var i = 0; i < nbPoints-1; i++) {
78        var squareDistance = 0;
79        for (var j = 0; j < 3; j++) {
80          squareDistance +=
81            Math.pow(this.controlPoints[i+1][j] - this.controlPoints[i][j],
                  2);
82        }
83        distances[i] = Math.sqrt(squareDistance);
84      }
85      // Make time interval proportional to distance between control points
86      var ts = [0, distances[0]];
87      for (var i = 2; i < nbPoints; i++) {
88        ts[i] = ts[i-1]+distances[i-1];
89      }
90      length = ts[ts.length - 1];
91      for (var i = 0; i < nbPoints; i++) {
92        ts[i] /= length;
93      }
94
95      //INTERPOLATION
96      // as [x y z] for each point
```

```
97      var derivatives = [];
98      // For start and ending points; normal to the surface
99      derivatives[0]=[];
100     derivatives[nbPoints-1] = [];
101     for (var i = 0; i < 3; i++) {
102       derivatives[0][i] = -this.controlPoints[0][i];
103       derivatives[nbPoints-1][i] = this.controlPoints[nbPoints-1][i];
104     }
105     // As for other derivatives, we use discrete approx
106     switch (this.tangents) {
107       case "incoming":
108         for (var i = 1; i < nbPoints-1; i++) {
109           derivatives[i] = [];
110           for (var j = 0; j < 3; j++) {
111             derivatives[i][j] =
112               this.controlPoints[i][j] - this.controlPoints[i-1][j];
113           }
114         }
115         break;
116       case "outgoing":
117         for (var i = 1; i < nbPoints-1; i++) {
118           derivatives[i]=[];
119           for (var j = 0; j < 3; j++) {
120             derivatives[i][j] =
121               this.controlPoints[i+1][j] - this.controlPoints[i][j];
122           }
123         }
124         break;
125       case "symmetric":
126         for (var i = 1; i < nbPoints-1; i++) {
127           derivatives[i] = [];
128           for (var j = 0; j < 3; j++) {
129             derivatives[i][j] =
130               this.controlPoints[i+1][j] - this.controlPoints[i-1][j];
131           }
132         }
133         break;
134       default:
135         console.error("'tangents' should be one of the following:\
136           'incoming', 'outgoing', 'symmetric'");
137     }
138     for (var i = 0; i < derivatives.length; i++) {
139       var squaredDerivativeNorm = 0;
140       for (var j = 0; j < 3; j++) {
141           squaredDerivativeNorm += Math.pow(derivatives[i][j], 2);
142       }
143       var derivativeVector = [];
144       for (var j = 0; j < 3; j++) {
145           derivativeVector[j] = (derivatives[i][j]
146             /Math.sqrt(squaredDerivativeNorm))*length;
147       }
148       derivatives[i] = derivativeVector;
149     }
150
```

```
151     // RETURN POLYNOMIALS
152     /*Function that returns 4x4 array with polynomials [a,b,c,d] for given
            ts (t),
153         control points (p) and derivatives (pd).
154         Values of a, b, c and d are found by solving the system
155          (t)= a + b[(t-ti)/(ti1-ti)] + c[(t-ti)/(ti1-ti)]^2 + d[(t-ti)/(
                ti1-ti)]^3
156         along with its derivative, being ti and ti1 t_i and t_(i+1).
157     */
158     function poly(t, p, pd) {
159       coef = [];
160       for (var i = 0; i < t.length-1; i++) {
161         coef[i] = [p[i],
162                 (t[i+1]-t[i]) * pd[i],
163                 3*p[i+1] - (t[i+1]-t[i])*pd[i+1] - 2*(t[i+1]-t[i])*pd[i] -
                        3*p[i],
164                 (t[i+1]-t[i])*pd[i+1] - 2*p[i+1] + (t[i+1]-t[i])*pd[i] +
                    2*p[i]
165         ];
166       }
167       return coef;
168     }
169     // Col extracts columns for matrices as array-of-arrays.
170     function col(matrix, column) {
171       var array = [];
172       for (var i = 0; i < matrix.length; i++) {
173         array[i] = matrix[i][column];
174       }
175       return array;
176     }
177     this.xpoly = poly(ts, col(this.controlPoints, 0), col(derivatives, 0))
            ;
178     this.ypoly = poly(ts, col(this.controlPoints, 1), col(derivatives, 1))
            ;
179     this.zpoly = poly(ts, col(this.controlPoints, 2), col(derivatives, 2))
            ;
180     this.ts = ts;
181     this.length = length;
182   },
183
184   interpolate: function(ts) {
185   /** @function interpolate
186     * @memberof FiberSource
187     * @param {Number[]|Number} ts List of "timesteps" (or a single)
            between 0 and 1.
188       From a ‘‘FiberSource‘‘, which is a continuous representation, to a
189       ‘‘Fiber‘‘, a discretization of the fiber trajectory.
190     * @return {array} The trajectory of the fiber, discretized over the
            provided
191       timesteps in an array-of-arrays form (N, 3)
192   */
193   // interp implements equation used for coefficients [a,b,c,d], described
          above.
194     function interp(coef, t, ti, ti1) {
```

```
195       factor = (t-ti) / (ti1-ti);
196       return (coef[0] + coef[1]*factor + coef[2]*Math.pow(factor,2) +
197               coef[3]*Math.pow(factor,3));
198     }
199     // Single value option
200     if (ts.constructor !== Array) {
201       var N = 1;
202       ts = [ts];
203     } else {
204         var N = ts.length;
205       }
206     var trajectory = [];
207     traj: for (var i = 0; i < N; i++) {
208       for (var j = 0; j < this.ts.length-1; j++) {
209         if ((ts[i] >= this.ts[j]) && (ts[i] <= this.ts[j+1])) {
210           break;
211         }
212         if (j == this.ts.length-2) {
213           console.error('Value '+i+' (being '+ts[i]+
214           ') is out of bounds ['+this.ts[0]+','
215           +this.ts[this.ts.length-1]+'].');
216           continue traj;
217         }
218       }
219       trajectory[i] = [];
220       trajectory[i][0] = interp(this.xpoly[j], ts[i], this.ts[j], this.ts[
              j+1]);
221       trajectory[i][1] = interp(this.ypoly[j], ts[i], this.ts[j], this.ts[
              j+1]);
222       trajectory[i][2] = interp(this.zpoly[j], ts[i], this.ts[j], this.ts[
              j+1]);
223       if (ts.constructor !== Array) {
224         trajectory = trajectory[0];
225       }
226     }
227       return trajectory;
228   },
229   addObserver: function(object) {
230     /** @function addObserver
231       * @memberof FiberSource
232       * @param {object} object Object to be added to <i>this.observers</i>
              array.
233       * @desc Add object to <i>this.observers</i> property
234     */
235     this.observers.push(object)
236   },
237   notify: function() {
238     /** @function notify
239       * @memberof FiberSource
240       * @desc Runs .refresh() method to all objects present in <i>this.
              observers</i> property. Renders.
241     */
242     for(var i = 0; i < this.observers.length; i++) {
243       this.observers[i].refresh();
```

```
244       }
245       render();
246     },
247     // setControlPoint changes a control point for this Fiber.
248     // inputs: n (position in controlPoints array) and x, y, z coordinates.
249     setControlPoint: function(n, axis, value) {
250       /** @function setControlPoint
251         * @memberof FiberSource
252         * @desc Sets a value to a given Control Point in a given Axis.
                Refreshes coefficients and notifies observers.
253         * @param {number} n Index in this.controlPoints of Control Point to
                be set.
254         * @param {string} axis Axis in which to apply the change 'x', 'y' or
                'z'.
255         * @param {number} value Value to set.
256       */
257       switch (axis) {
258         case 'x':
259           this.controlPoints[n][0] = value;
260           break;
261         case 'y':
262           this.controlPoints[n][1] = value;
263           break;
264         case 'z':
265           this.controlPoints[n][2] = value;
266           break;
267         default: console.error('Wrong axis set in fiber.setControlPoint.
              Called for ' + axis);
268       }
269       this.polyCalc();
270       this.notify();
271     }
272   }
273
274   function IsotropicRegionSource(center, radius, color) {
275   /** @class IsotropicRegionSource
276     * @classdesc An Isotropic Region is defined in Phantomas as an empty
          spherical space.<br>
277     * IsotropicRegionSource is the basic Class for the representation of an
          Isotropic Region. Objects containing
278     the geometries to be added to the scene are to be referred to
          IsotropicRegionSource for
279     gathering any necessary information.
280
281     * @param {array} center Array containing the 3-D coordinates
282     of the center point in which the Isotropic Region is located.
283     * @param {number} [radius=1] Isotropic Region radius.
284     * @param {number} [color] Color in which the Isotropic Region should be
          displayed in. If not
285     specified, to be picked randomly from {@link colors}.
286
287     * @property {array} observers Objects to be notified when some change is
           applied
288   */
```

```
289   this.center = center;
290   this.radius = radius;
291
292   if (color === undefined) {
293     color = new THREE.Color(colors[Math.floor(Math.random()*colors.length)
          ]);
294   }
295   this.color = new THREE.Color(color);
296
297   this.observers = [];
298 }
299 IsotropicRegionSource.prototype = {
300   // Refreshes objects in the observer list
301   notify: function() {
302     /** @function notify
303       * @memberof IsotropicRegionSource
304       * @desc Runs .refresh() method to all objects present in <i>this.
            observers</i> property. Renders.
305     */
306
307     for(var i = 0; i < this.observers.length; i++) {
308       this.observers[i].refresh();
309     }
310     render();
311   },
312   setCenter: function(axis, value) {
313   /** @function setCenter
314     * @memberof IsotropicRegionSource
315     * @param {string} axis Axis in which to apply the new value
316     * @param {number} value New value to be applied
317     * @desc Changes the center of the Isotropic Region for a given new.
          Notifies observers.
318   */
319     switch (axis) {
320       case 'x':
321         this.center[0] = value;
322         break;
323       case 'y':
324         this.center[1] = value;
325         break;
326       case 'z':
327         this.center[2] = value;
328         break;
329       default: console.error('Incorrect axis label for IsotropicRegion.
            setCenter(label, value)');
330     }
331     this.notify();
332   },
333   setRadius: function(radius) {
334     /** @function setRadius
335       * @memberof IsotropicRegionSource
336       * @param {number} radius New value to be applied
337       * @desc Changes the radius of the Isotropic Region for a given new.
            Notifies observers.
```

```
338       */
339       this.radius = radius;
340       this.notify();
341     },
342     addObserver: function(object) {
343       /** @function addObserver
344         * @memberof IsotropicRegionSource
345         * @param {object} object Object to be added to <i>this.observers</i>
                array.
346         * @desc Add object to <i>this.observers</i> property
347       */
348
349       this.observers.push(object)
350     }
351   }
```

2.3.16 MeshSource.js

```
1   /**@overview Contains Class definitions for {@link FiberSkeleton}, {@link
        FiberTube} and {@link IsotropicRegion}.*/
2
3   function FiberSkeleton(fiber, parameters) {
4   /** @class FiberSkeleton
5     * @classdesc FiberSkeleton creates 3D representation of control points
          and fiber path
6       from a given fiber.<br>
7       Subject-Observer pattern must be enabled from its {@link FiberSource}
            reference and fired from subject with {@link FiberSkeleton.refresh
            |refresh();}.
8     * @property {THREE.Line} line The thread representing the path. Ready
          for {@link scene}.add.
9     * @property {THREE.Mesh} spheres Big mesh containing all control-point
          marking spheres. Ready for {@link scene}.add.
10    * @property {FiberSource} fiber Reference to source fiber object.
11    * @property {THREE.Color} color Color of the thread.
12    * @property {Number} sphereSegments Amount of segments (in each
          dimension) each controlpoint sphere will feature.
13    * @property {Number} lineSegments Amount of segments the thread will
          feature.
14    *
15    * @param {FiberSource} fiber Reference fiber.
16    * @param {Object} [parameters] Optional parameters
17    * @param {Number} [parameters.lineSegments=256] Number of axial segments
           to feature in line's {@link FiberSkeleton}
18    * @param {Number} [parameters.sphereSegments=32] Number of radial
          segments to feature in each control point of {@link FiberSkeleton}
19    * @param {THREE.Color} [parameters.color] Color of the thread. If not
          specified, generated randomly from {@link colors}.
20    */
21    this.fiber = fiber;
22    this.points = fiber.controlPoints; //Private property
23
24    // Assign either specified parameters or default values
25    if (!parameters) var parameters = [];
```

```
26    if (parameters.color === undefined) {
27      this.color = fiber.color;
28    } else {
29      this.color = parameters.color;
30    }
31    if (parameters.lineSegments === undefined) {
32      this.lineSegments = 256;
33    } else {
34      this.lineSegments = parameters.lineSegments;
35    }
36    if (parameters.sphereSegments === undefined) {
37      this.sphereSegments = 32;
38    } else {
39      this.sphereSegments = parameters.sphereSegments;
40    }
41
42    // Create line thread
43    // Interpolate points for THREE.BufferAttribute needs
44    discretePoints = new Float32Array(3*this.lineSegments+3);
45    for (var i = 0; i <= this.lineSegments; i++) {
46      discretePoints.set([fiber.interpolate(i/this.lineSegments)[0][0],
47                          fiber.interpolate(i/this.lineSegments)[0][1],
48                          fiber.interpolate(i/this.lineSegments)[0][2]], 3*
                                i);
49    }
50    // Create trajectory
51    var trajectory = new THREE.BufferGeometry();
52    trajectory.addAttribute('position', new THREE.BufferAttribute(
          discretePoints, 3));
53    // Create thread material
54    var thread = new THREE.LineBasicMaterial({ color:this.color, linewidth:
          1 });
55    // / Build line
56    this.line = new THREE.Line(trajectory, thread);
57
58    // Create sphere mesh for controlPoints
59    // sphere is the prototype sphere
60    var sphere = new THREE.SphereGeometry(fiber.radius/5, this.
          sphereSegments, this.sphereSegments);
61    var sphereGeometry = new THREE.Geometry();
62    // All spheres to be added are meshed in one single geometry
63    var meshes = [];
64    for (var i = 0; i < this.points.length; i++) {
65      meshes[i] = new THREE.Mesh(sphere);
66      meshes[i].position.set(this.points[i][0], this.points[i][1], this.
            points[i][2]);
67      meshes[i].updateMatrix();
68      sphereGeometry.merge(meshes[i].geometry, meshes[i].matrix);
69    }
70    var surface = new THREE.MeshBasicMaterial( {color: 0xffff00} );
71    // Build spheres mesh
72    this.spheres = new THREE.Mesh(sphereGeometry, surface);
73  }
74
```

```
75  FiberSkeleton.prototype.refresh = function() {
76    /** @function refresh
77      * @memberof FiberSkeleton
78      * @desc Updates thread and spheres position from self properties. Must
            be fired when those change.
79      */
80    // Spheres mesh must be built again
81    var sphere = new THREE.SphereGeometry(this.fiber.radius/5, this.
          sphereSegments, this.sphereSegments);
82    var sphereGeometry = new THREE.Geometry();
83    var meshes = [];
84    for (var i = 0; i < this.points.length; i++) {
85      meshes[i] = new THREE.Mesh(sphere);
86      meshes[i].position.set(this.points[i][0], this.points[i][1], this.
            points[i][2]);
87      meshes[i].updateMatrix();
88      sphereGeometry.merge(meshes[i].geometry, meshes[i].matrix);
89    }
90    this.spheres.geometry = sphereGeometry;
91
92    // Thread trajectory must be built again as well
93    var discretePoints = new Float32Array(3*this.lineSegments+3);
94    for (var i = 0; i <= this.lineSegments; i++) {
95      discretePoints.set([this.fiber.interpolate(i/this.lineSegments)[0][0],
96                          this.fiber.interpolate(i/this.lineSegments)
                                [0][1],
97                          this.fiber.interpolate(i/this.lineSegments)
                                [0][2]], 3*i);
98    }
99    var trajectory = new THREE.BufferGeometry();
100   trajectory.addAttribute('position', new THREE.BufferAttribute(
          discretePoints, 3));
101   this.line.geometry = trajectory;
102   // Line color is to be kept
103   this.line.material.color = this.color;
104 }
105
106 function FiberTube(fiber, parameters) {
107   /** @class FiberTube
108     * @classdesc FiberTube creates a 3D representation of a given fiber in
            a tubular shape
109     of given radius.
110     <br>Subject-Observer pattern must be enabled from its {@link
            FiberSource} reference and fired from subject with {@link
            FiberSkeleton.refresh|refresh();}.
111     * @property {THREE.Mesh} mesh Mesh containing the fiber's tubular
            shape. Ready for {@link scene}.add.
112     * @property {FiberSource} fiber Reference to source fiber object.
113     * @property {THREE.Color} color Color of the tube.
114     * @property {Number} axialSegments Amount of segments to feature in
            the tube mesh.
115     * @property {Number} radialSegments Amount of radial segments to
            feature in the tube mesh.
116     *
```

```
117     * @param {FiberSource} fiber Reference fiber.
118     * @param {Object} [parameters] Optional parameters
119     * @param {Number} [parameters.axialSegments=256] Number of axial
            segments to feature in the tube mesh.
120     * @param {Number} [parameters.radialSegments=64] Number of radial
            segments to feature in the tube mesh.
121     * @param {THREE.Color} [parameters.color] Color of the thread. If not
            specified, generated randomly from {@link colors}.
122     */
123
124   this.fiber = fiber;
125   // Check if radius was specified. If not, set default.
126   if (this.fiber.radius === undefined) this.fiber.radius = 1;
127   var radius = this.fiber.radius;
128   // Assign either specified parameters or default values
129   if (!parameters) var parameters = [];
130   if (parameters.color === undefined) {
131     this.color = fiber.color;
132   } else {
133     this.color = parameters.color;
134   }
135   if (parameters.axialSegments === undefined) {
136     this.axialSegments = 256;
137   } else {
138     this.axialSegments = parameters.axialSegments;
139   }
140   if (parameters.radialSegments === undefined) {
141     this.radialSegments = 64;
142   } else {
143     this.radialSegments = parameters.radialSegments;
144   }
145
146   //Curve is part of tube geometry
147   this.curve = Object.create(THREE.Curve.prototype); //Private property
148   // .getPoint must be part of curve object
149   this.curve.getPoint = function(t) {
150       var tx = fiber.interpolate(t)[0][0];
151       var ty = fiber.interpolate(t)[0][1];
152       var tz = fiber.interpolate(t)[0][2];
153   return new THREE.Vector3(tx, ty, tz);
154   }
155   // Geometry and materials are created
156   var geometry = new THREE.TubeGeometry(this.curve,
157                         this.axialSegments , radius, this.radialSegments);
158   var material = new THREE.MeshPhongMaterial( { color:this.color, shading:
          THREE.FlatShading } );
159   // Transparency must be enabled so to be able to fade the tube
160   material.transparent = true;
161   // Double side is needed so a tube appareance is acquired
162   material.side = THREE.DoubleSide;
163   this.mesh = new THREE.Mesh(geometry, material);
164 }
165 FiberTube.prototype.refresh = function() {
166   /** @function refresh
```

```
167     * @memberof FiberTube
168     * @desc Updates tube shape and position from self properties. Must be
            fired when those change.
169     */
170   this.mesh.geometry = new THREE.TubeGeometry(this.curve,
171                 this.axialSegments, this.fiber.radius, this.radialSegments
                        );
172 }
173
174 /* IsotropicRegion creates a mesh from an IsotropicRegionSource.
175   Subject-Observer pattern must be enabled and fired from subject with .
          refresh();
176   Input: source - IsotropicRegionSource
177           parameters object (not compulsory):
178            .color
179            .widthSegments
180            .heightSegments
181
182
183   Main properties:
184     .mesh - the mesh of the region ready for scene.add.
185
186   Other properties:
187     .source - The region from which the representation constructed (
            IsotropicRegionSource)
188     .widthSegments and .heightSegments: The segments that make up the
            sphere in
189        each dimension. Default is 128.
190     .color - color of the sphere
191
192    Methods:
193     .refresh() - Updates mesh after source change. No input no output.
194 */
195 function IsotropicRegion(source, parameters) {
196   /** @class IsotropicRegion
197     * @classdesc IsotropicRegion creates a 3D representation of an
            Isotropic Region.
198     <br>Subject-Observer pattern must be enabled from its {@link
            FiberSource} reference and fired from subject with {@link
            FiberSkeleton.refresh|refresh();}.
199     * @property {THREE.Mesh} mesh Mesh containing the Isotropic Region
            sphere. Ready for {@link scene}.add.
200     * @property {IsotropicRegionSource} fiber Reference to source fiber
            object.
201     * @property {THREE.Color} color Color of the sphere.
202     * @property {Number} widthSegments Amount of width segments to feature
             in the sphere mesh.
203     * @property {Number} heightSegments Amount of height segments to
            feature in the sphere mesh.
204     *
205     * @param {IsotropicRegionSource} source Reference Isotropic Region.
206     * @param {Object} [parameters] Optional parameters
207     * @param {Number} [parameters.widthSegments=128] Number of width
            segments to feature in the sphere mesh.
```

```
208     * @param {Number} [parameters.heightSegments=128] Number of height
            segments to feature in the sphere mesh.
209     * @param {THREE.Color} [parameters.color] Color of the thread. If not
            specified, generated randomly from {@link colors}.
210     */
211
212   this.source = source;
213
214   // Assign either specified parameters or default values
215   if (!parameters) var parameters = [];
216   if (parameters.color === undefined) {
217     this.color = source.color;
218   } else {
219     this.color = parameters.color;
220   }
221   if (parameters.widthSegments === undefined) {
222     this.widthSegments = 128;
223   } else {
224     this.widthSegments = parameters.widthSegments;
225   }
226   if (parameters.heightSegments === undefined) {
227     this.heightSegments = 128;
228   } else {
229     this.heightSegments = parameters.heightSegments;
230   }
231
232   var geometry = new THREE.SphereGeometry( source.radius, this.
         widthSegments, this.heightSegments );
233   var material = new THREE.MeshPhongMaterial( { color:this.color, shading:
          THREE.FlatShading } );
234   // Transparency must be enabled so to be able to fade the tube
235   material.transparent = true;
236   this.mesh = new THREE.Mesh(geometry, material);
237   this.mesh.position.set(source.center[0], source.center[1], source.center
         [2]);
238 }
239 IsotropicRegion.prototype.refresh = function() {
240   /** @function refresh
241     * @memberof IsotropicRegion
242     * @desc Updates sphere position and radius from self properties. Must
            be fired when those change.
243     */
244     this.mesh.geometry = new THREE.SphereGeometry( this.source.radius,
            this.widthSegments, this.heightSegments );
245     this.mesh.position.set(this.source.center[0], this.source.center[1],
            this.source.center[2]);
246 }
```

2.3.17 axes.js

```
1  function buildAxes( length ) {
2  /** @function buildAxes
3    * @desc Returns axes to be added to the scene.
4    * @author [Soledad Penades]{@link https://soledadpenades.com/articles/
```

```
      three-js-tutorials/drawing-the-coordinate-axes/}
  * @param {number} length The axis radius
  * @return The axis geometry ready to be added to the scene
*/
  var axes = new THREE.Object3D();
  var linewidth = 2;

  function buildAxis( src, dst, colorHex, dashed ) {
    var geom = new THREE.Geometry(),
        mat;

    if(dashed) {
            mat = new THREE.LineDashedMaterial({ linewidth: linewidth,
                color: colorHex, dashSize: 3, gapSize: 3 });
    } else {
            mat = new THREE.LineBasicMaterial({ linewidth: linewidth,
                color: colorHex });
    }

    geom.vertices.push( src.clone() );
    geom.vertices.push( dst.clone() );
    geom.computeLineDistances(); // This one is SUPER important, otherwise
         dashed lines will appear as simple plain lines

    var axis = new THREE.Line( geom, mat, THREE.LineSegments );

    return axis;
  }

  axes.add( buildAxis( new THREE.Vector3( 0, 0, 0 ), new THREE.Vector3(
      length, 0, 0 ), 0xFF0000, false ) ); // +X
  axes.add( buildAxis( new THREE.Vector3( 0, 0, 0 ), new THREE.Vector3( -
      length, 0, 0 ), 0xFF0000, true) ); // -X
  axes.add( buildAxis( new THREE.Vector3( 0, 0, 0 ), new THREE.Vector3( 0,
       length, 0 ), 0x00FF00, false ) ); // +Y
  axes.add( buildAxis( new THREE.Vector3( 0, 0, 0 ), new THREE.Vector3( 0,
       -length, 0 ), 0x00FF00, true ) ); // -Y
  axes.add( buildAxis( new THREE.Vector3( 0, 0, 0 ), new THREE.Vector3( 0,
       0, length ), 0x0000FF, false ) ); // +Z
  axes.add( buildAxis( new THREE.Vector3( 0, 0, 0 ), new THREE.Vector3( 0,
       0, -length ), 0x0000FF, true ) ); // -Z

  return axes;
}
```

2.3.18 load.js

```
/** @function loadPhantom
  * @desc Loads a Phantom contained in a JSON file and puts it into the
       scene.<br>
  * Contains all the functions necessary to translate Phantomas' JSON
       structure
  * to {@link FiberSource} parameters.
  * @param {String} string
```

```
6     * The string containing the parsed phantom variable.
7     * @return {Phantom} The phantom ready to be added to the scene.
8   */
9   function loadPhantom( string ) {
10    var phantom = new Phantom();
11    var loadedFibers = JSON.parse(string).fiber_geometries;
12    var loadedRegions = JSON.parse(string).isotropic_regions;
13    // Send number of elements as parameters to addFiber and
          addIsotropicRegion
14    nbLoads = 0;
15    if (loadedFibers) nbLoads += Object.keys(loadedFibers).length;
16    if (loadedRegions) nbLoads += Object.keys(loadedRegions).length;
17    var parameters = {
18      nbElements: nbLoads
19    };
20
21    // Objects will be added in Phantom
22    for (var property in loadedFibers) {
23      if (loadedFibers.hasOwnProperty(property)) {
24        var fiber = loadedFibers[property.toString()];
25        // Control points need to be transleted to array-of-arrays form
26        var newcp = [];
27        for (var i = 0; i < fiber.control_points.length; i = i + 3) {
28          newcp.push([roundToPrecision(fiber.control_points[i]),
29                      roundToPrecision(fiber.control_points[i+1]),
30                      roundToPrecision(fiber.control_points[i+2])]);
31        }
32        phantom.addFiber(new FiberSource(newcp, fiber.tangents,
              roundToPrecision(fiber.radius), fiber.color), parameters);
33      }
34    }
35    for (var property in loadedRegions) {
36      if (loadedRegions.hasOwnProperty(property)) {
37        var region = loadedRegions[property.toString()];
38        phantom.addIsotropicRegion(new IsotropicRegionSource(region.center,
              roundToPrecision(region.radius), region.color), parameters);
39      }
40    }
41
42    // log an error in case fibers or regions were not found.
43    if (phantom.isotropicRegions.source.length == 0) console.warn('Any
          region found in file');
44    if (phantom.fibers.source.length == 0) console.warn('Any fiber found in
          file');
45    return phantom;
46  }
```

2.3.19 main.js

```
1   /** @overview Main file. Contains {@link init}, {@link render}, {@link
        animate} and {@link show} functions, as well as main global functions
        and constants.
2   */
3   var mesh, renderer, scene, camera, directionalLight, controls, phantom,
```

```
    dragAndDrop;
var meshConstraints = {
  /** @constant {object} meshConstraints Constant used in {@link Phantom.
      addFiber} and {@link Phantom.addIsotropicRegion} for defining
      segments in meshes.
  Used as a parameter for WEBGL stability.
  * @property {Number} maxTotalAxialSegments Maximum number of Axial
      Segments in fiber tube mesh to appear in the scene
  * @property {Number} maxMeshAxialSegments Maximum number of Axial
      Segments to feature in a fiber tube mesh
  * @property {Number} maxTotalRadialSegments Maximum number of Radial
      Segments in fiber tube mesh to appear in the scene
  * @property {Number} maxMeshRadialSegments Maximum number of Radial
      Segments to feature in a fiber tube mesh
  * @property {Number} maxTotalLineSegments Maximum number of Line
      Segments to appear in the scene
  * @property {Number} maxMeshLineSegments Maximum number of Line Segments
       to feature in a single skeleton line
  * @property {Number} maxTotalSkeletonSphereSegments Maximum number of
      Radial Segments in skeleton control points to appear in the scene
  * @property {Number} maxMeshSkeletonSphereSegments Maximum number of
      Radial Segments to feature in a single control point sphere mesh
  * @property {Number} maxTotalIsotropicRegionSegments Maximum number of
      Radial Segments in Isotropic Regions to appear in the scene
  * @property {Number} maxMeshIsotropicRegionSegments Maximum number of
      Radial Segments in Isotropic Regions to appear in a single mesh
  */
    maxTotalAxialSegments: 1440,
    maxMeshAxialSegments: 64,

    maxTotalRadialSegments: 480,
    maxMeshRadialSegments: 32,

    maxTotalLineSegments: 960,
    maxMeshLineSegments: 128,

    maxTotalSkeletonSphereSegments: 240,
    maxMeshSkeletonSphereSegments: 16,

    maxTotalIsotropicRegionSegments: 1024,
    maxMeshIsotropicRegionSegments: 32
}

/** @constant {Number} precision
  * @desc Number of decimal digits present in all values given. Used in {
      @link roundToPrecision}.
*/
var precision = 1; // in amount of decimal digits
function roundToPrecision(number) {
/** @function roundToPrecision
  * @desc Applies precision to any value. Uses {@link precision} constant.
  * @param {Number|String} number Number to round
  * @return {Number} The rounded number
  */
```

```
44
45    // Correct type
46    number = Number(number);
47
48    number *= 10 * precision;
49    number = Math.round(number);
50    number /= 10 * precision;
51    return number;
52  }
53
54
55  window.onload = init;
56
57
58  function render() {
59  /** @function render
60    * @desc Renders the scene. Must be called anytime the scene is changed.
61    */
62    renderer.render(scene, camera);
63  }
64
65  function animate() {
66  /** @function animate
67    * @desc Called recursively. Updates the controls as well.
68    */
69    requestAnimationFrame( animate );
70    controls.update();
71  }
72  function init() {
73  /** @function init
74    * @desc To be called when page is loaded, initialises the app. Starting
          from a XMLHttpRequest,
75    calls [show()]{@link show} and [setupGUI()]{@link module:GUI
          Construction.setupGUI}.
76    */
77
78    if (location.href.indexOf('?') > 0) {
79      path = location.href.substring( location.href.indexOf('?') + 1);
80      makeRequest();
81    } else {
82      phantom = new Phantom();
83      console.log('No specified path found. Loading scratch mode.');
84      show();
85      setupGUI();
86    }
87
88    function makeRequest() {
89      var request = new XMLHttpRequest();
90      request.overrideMimeType("text/plain");
91      request.open("get", path, true);
92      request.onreadystatechange = function() {
93        if (this.readyState === 4) {
94          if (this.status === 200) {
95            phantom = loadPhantom(this.response);
```

```
96          } else {
97            console.error('Error: ' + path + ' was not found. Loading
                  scratch mode.')
98            phantom = new Phantom();
99          }
100         show();
101         setupGUI();
102       }
103     };
104     request.send(null);
105   }
106 }
107 /** @function show
108   * @desc Initialises everything regarding the THREE.js environtment. Adds
          window events.
109   * @requires THREE.js
110   * @requires TrackballControls.js
111 */
112 function show() { // The rendering engine is initialized
113   renderer = new THREE.WebGLRenderer();
114   var container = document.getElementById('container');
115
116   renderer.setSize(container.offsetWidth, window.innerHeight);
117   // It is appended to the div container in the HTML5 tree
118   container.appendChild(renderer.domElement);
119
120   // We create a scene and a camera. Position is to be corrected further
          in the code.
121   camera = new THREE.PerspectiveCamera(50,
122                                        container.offsetWidth / (window.
                                             innerHeight),
123                                        1,
124                                        10000);
125   camera.position.set(0, 0, 0);
126
127   // Create, the scene and add cameras, lights.
128   scene = new THREE.Scene();
129   scene.add(camera);
130   scene.add(new THREE.AmbientLight( 0xffffff, .5 ) );
131   // Directional lights are added in all directions
132   for (var x = -100; x <= 100; x=x+200) {
133     for (var y = -100; y <= 100; y=y+200) {
134       for (var z = -100; z <= 100; z=z+20) {
135         var directionalLight = new THREE.DirectionalLight(0x555555, .15);
136         directionalLight.position.x = x;
137         directionalLight.position.y = y;
138         directionalLight.position.z = z;
139         scene.add(directionalLight);
140       }
141     }
142   }
143
144   // Load phantom and add it in the scene
145   phantom.addToScene(scene);
```

```
146
147   camera.position.z = phantom.radius() * 2 * 1.5;
148
149   renderer.render(scene, camera);
150
151   // Add mouse control to the camera
152   controls = new THREE.TrackballControls( camera , renderer.domElement );
153   controls.enableZoom = true;
154   controls.rotateSpeed = 2.5;
155   controls.zoomSpeed = 1;
156   controls.noPan = false;
157   controls.addEventListener('change', render);
158
159   // Leave confirmation. From https://stackoverflow.com/questions
          /10311341/confirmation-before-closing-of-tab-browser
160   window.onbeforeunload = function (e) {
161     e = e || window.event;
162
163     // For IE and Firefox prior to version 4
164     if (e) {
165         e.returnValue = 'Leave Phantomas?';
166     }
167
168     // For Safari
169     return 'Leave Phantomas?';
170   };
171   window.addEventListener( 'resize', onWindowResize, false );
172   function onWindowResize(){
173     camera.aspect = container.offsetWidth / (window.innerHeight);
174     camera.updateProjectionMatrix();
175
176     renderer.setSize( container.offsetWidth, window.innerHeight);
177     render();
178
179     resizeGUI();
180   }
181   // Call animate to start the recursive call.
182   animate();
183 }
```

2.3.20 Phantom.js

```
1  /** @overview Contains {@link Phantom} definition and added methods to {
       @link THREE.Scene} prototype.*/
2  /** @module THREE*/
3
4  // Adding a scene method that removes all phantoms present.
5  // This way, cameras and lights are never removed.
6  /** @class Scene
7    * @memberof module:THREE
8    * @classdesc THREE.js class for a Scene. {@link https://threejs.org/docs
         /index.html?q=sce#Reference/Scenes/Scene|Link to THREE.js
         documentation}
9    */
```

```
10 THREE.Scene.prototype.removePhantom = function() {
11 /** @method removePhantom
12   * @memberof module:THREE.Scene
13   * @desc Removes all Phantoms present in the Scene, leaving everything
       not being part of a Phantom.
14   */
15   var objects = [];
16   this.children.forEach( function(object){
17     if (((object.type == 'Mesh') || (object.type == "Line")) && (!object.
         isHighlight)) {
18       objects.push(object);
19     }
20   });
21
22   var scene = this;
23   // If removed inside scene's forEach, length is changed and so algorithm
       may skip some objects removal.
24   objects.forEach( function(object) {
25     scene.remove(object);
26   });
27
28   this.removeCPHighlight(true);
29 }
30
31 // All boolean makes red points to be removed if true.
32 THREE.Scene.prototype.removeCPHighlight = function(all) {
33   /** @method removeCPHighlight
34     * @memberof module:THREE.Scene
35     * @desc Removes Control Point highlights. By default, only blue
         colored highlights, used
36     * when hover.
37     * @param {boolean} [all] If true, removes red colored highlight as
         well.
38     */
39   var objects = [];
40   this.children.forEach( function(object){
41     if (object.isHighlight) {
42       if (object.isBlueHighlight) {
43         objects.push(object);
44       } else if (all) {
45         objects.push(object);
46       }
47     }
48   });
49
50   var scene = this;
51   // If removed inside scene's forEach, length is changed and so algorithm
       may skip some objects removal.
52   objects.forEach( function(object) {
53     scene.remove(object);
54   });
55   render();
56 }
57
```

```
58 function Phantom() {
59   /** @class Phantom
60     * @global
61     * @classdesc Includes all data regarding a Phantom and methods to
           modify its appareance.
62
63     * @property {Object} fibers Contains all objects for Fiber definition
           and representation.
64     * @property {FiberSource[]} fibers.source FiberSource for each fiber
           bundle.
65     * @property {FiberTube[]}  fibers.tube FiberTube for each fiber bundle
           . Must have same index as <i>source</i>
66     * @property {FiberSkeleton[]}  fibers.skeleton FiberSkeleton for each
           fiber bundle. Must have same index as <i>source</i>
67
68     * @property {Object} isotropicRegions Contains all objects for
           Isotropic Region definition and representation.
69     * @property {IsotropicRegionSource[]} isotropicRegions.source
           IsotropicRegionSource for each region bundle.
70     * @property {IsotropicRegion[]}  isotropicRegions.sphere
           IsotropicRegion for each region bundle. Must have same index as <i
           >source</i>
71
72     * @property {Number} highlightOpacity=0.1 The base opacity for fading
           meshes in the scene. Over 1.
73     * @property {THREE.Color} highlightColor=null The color to be taken by
            highlighted objects. By default, <i>null</i>, thus color does not
            change.
74     */
75   this.fibers = {
76     source: [],
77     tube: [],
78     skeleton: []
79   }
80   this.isotropicRegions = {
81     source: [],
82     sphere: []
83   }
84   this.highlightOpacity = 0.1;
85   this.highlightColor = null;
86 }
87
88 Phantom.prototype = {
89   addFiber: function(fiber, parameters, replaceindex) {
90   /** @function addFiber
91     * @memberof Phantom
92     * @desc Adds a Fiber to the Phantom, by creating their {@link
           FiberTube} and {@link FiberSkeleton}.
93     * @param {FiberSource} fiber Fiber to be added to the Phantom.
94     * @param {Object} [parameters] Optional object containing different
           optional parameters. Those parameters
95     may be the ones to be specified in classes {@link FiberTube} and {
           @link FiberSkeleton} constructor.
96     <br>If only nbElements is defined, sets the rest of those parameters
```

```
        by itself, looking at the
97      constant {@link meshConstraints}.
98      * @param {Number} [parameters.nbElements] Number of elements to
            feature in the whole Phantom.
99      This allows the function to limit by itself the amout of segments
            present in the scene. Configurable via constant {@link
            meshConstraints}. <br>Only taken into
100     account if any parameter regarding segments is defined.
101     * @param {Number} [replaceindex] If specified, replaces the Fiber yet
            present in this same index.
102     */
103
104     /* If not specified, set segments constraint so renderer is stable in
            browser
105     This grabs nbElements thrown by load function and sets the number of
            segments
106     each mesh will feature, taking values from global variable
            meshConstraints (main.js) */
107     if (!parameters) var parameters = [];
108     if ((parameters.nbElements) && (!parameters.axialSegments) && (!
            parameters.radialSegments)) {
109       parameters.axialSegments = Math.min(Math.floor(meshConstraints.
              maxTotalAxialSegments / parameters.nbElements), meshConstraints.
              maxMeshAxialSegments);
110       parameters.radialSegments = Math.min(Math.floor(meshConstraints.
              maxTotalRadialSegments / parameters.nbElements), meshConstraints
              .maxMeshRadialSegments);
111     }
112     if ((parameters.nbElements) && (!parameters.lineSegments) && (!
            parameters.sphereSegments)) {
113       parameters.axialSegments = Math.min(Math.floor(meshConstraints.
              maxTotalLineSegments / parameters.nbElements), meshConstraints.
              maxMeshLineSegments);
114       parameters.sphereSegments = Math.min(Math.floor(meshConstraints.
              maxTotalSkeletonSphereSegments / parameters.nbElements),
              meshConstraints.maxTotalSkeletonSphereSegments);
115     }
116
117     if (replaceindex !== undefined) {
118
119       this.fibers.source[replaceindex] = new FiberSource(fiber.
              controlPoints, fiber.tangents, fiber.radius, fiber.color);
120       this.fibers.skeleton[replaceindex] = new FiberSkeleton(this.fibers.
              source[replaceindex], parameters);
121       this.fibers.tube[replaceindex] = new FiberTube(this.fibers.source[
              replaceindex], parameters);
122
123       this.fibers.source[replaceindex].addObserver(this.fibers.tube[
              replaceindex]);
124       this.fibers.source[replaceindex].addObserver(this.fibers.skeleton[
              replaceindex]);
125
126     } else {
127
```

```
128       this.fibers.source.push(fiber);
129       this.fibers.skeleton.push(new FiberSkeleton(fiber, parameters));
130       this.fibers.tube.push(new FiberTube(fiber, parameters));
131       parameters.color = undefined;
132       // Skeleton and Tube added as observers.
133       fiber.addObserver(this.fibers.tube[this.fibers.tube.length-1]);
134       fiber.addObserver(this.fibers.skeleton[this.fibers.skeleton.length
              -1]);
135
136     }
137
138   },
139   addCP: function(fiberindex, cpbefore) {
140   /** @function addCP
141     * @memberof Phantom
142     * @desc Adds a new Control Point to a specified Fiber in the Phantom.
143     <br>No position is specified; control point is added over the
            trajectory in between two existing control points.
144     <br>Will {@link render}.
145     * @param {Number} fiberindex Index of the fiber in which the control
            point is to be added to.
146     * @param {Number} cpbefore Index of the control point in which the new
             one is to be added after.
147     */
148     var fiber = this.fibers.source[fiberindex];
149     var newts = (fiber.ts[cpbefore] + fiber.ts[cpbefore + 1]) / 2;
150     var newcp = fiber.interpolate(newts)[0];
151     newcp.forEach( function(coordinate, index) {
152       newcp[index] = roundToPrecision(coordinate);
153     });
154     fiber.controlPoints.splice(cpbefore + 1, 0, newcp);
155
156     var parameters = {
157       nbElements: this.fibers.source.length + this.isotropicRegions.source
              .length
158     };
159     phantom.addFiber(fiber, parameters, fiberindex);
160   },
161   removeCP: function(fiberindex, cp) {
162     /** @function removeCP
163       * @memberof Phantom
164       * @desc Removes an existing Control Point of a specified Fiber in
              the Phantom.
165       <br>Will {@link render}.
166       * @param {Number} fiberindex Index of the fiber in which the control
               point is to be removed.
167       * @param {Number} cp Index of the control point to be removed.
168       */
169     var fiber = this.fibers.source[fiberindex];
170     fiber.controlPoints.splice(cp, 1);
171     var parameters = {
172       nbElements: this.fibers.source.length + this.isotropicRegions.source
              .length
173     };
```

```
174     phantom.addFiber(fiber, parameters, fiberindex);
175   },
176   addIsotropicRegion: function(region, parameters) {
177     /** @function addIsotropicRegion
178       * @memberof Phantom
179       * @desc Adds a Fiber to the Phantom, by creating their {@link
              IsotropicRegion}.
180       * @param {IsotropicRegionSource} region Fiber to be added to the
              Phantom.
181       * @param {Object} [parameters] Optional object containing different
              optional parameters. Those parameters
182       may be the ones to be specified in class {@link IsotropicRegion}
              constructor.
183       <br>If only nbElements is defined, sets the rest of those parameters
               by itself, looking at the
184       constant {@link meshConstraints}.
185       * @param {Number} [parameters.nbElements] Number of elements to
              feature in the whole Phantom.
186       This allows the function to limit by itself the amout of segments
              present in the scene. Configurable via constant {@link
              meshConstraints}. <br>Only taken into
187       account if any parameter regarding segments is defined.
188       */
189
190     /* If not specified, set segments constrainct so renderer is stable in
            browser
191     This grabs nbElements thrown by load function and sets the number of
          segments
192     each mesh will feature, from global variable meshConstraints (main.js)
           */
193     if (!parameters) var parameters = [];
194     if ((parameters.nbElements) && (!parameters.heightSegments) && (!
          parameters.widthSegments)) {
195       parameters.heightSegments = Math.min(Math.floor(meshConstraints.
              maxTotalIsotropicRegionSegments / parameters.nbElements),
              meshConstraints.maxMeshIsotropicRegionSegments);
196       parameters.widthSegments = Math.min(Math.floor(meshConstraints.
              maxTotalIsotropicRegionSegments / parameters.nbElements),
              meshConstraints.maxMeshIsotropicRegionSegments);
197     }
198
199     this.isotropicRegions.source.push(region);
200     this.isotropicRegions.sphere.push(new IsotropicRegion(region,
          parameters));
201
202     region.addObserver(this.isotropicRegions.sphere[this.isotropicRegions.
          sphere.length-1]);
203   },
204   radius: function() {
205     /** @function radius
206       * @memberof Phantom
207       * @desc Provides the radius of the Phantom, understood as the
              distance from the center
208       to the most distant bundle.
```

```
209       * @returns {Number} Radius of the Phantom.
210       */
211     var maxdist = 0;
212     // Return is the farthest point from the center
213     for (var i = 0; i < this.fibers.source.length; i++) {
214       var cp = this.fibers.source[i].controlPoints;
215       for (var j = 0; j < cp.length; j++) {
216         var dist = Math.sqrt(
217           Math.pow(cp[j][0],2) +
218           Math.pow(cp[j][1],2) +
219           Math.pow(cp[j][2],2)
220         );
221         if (dist > maxdist) {maxdist = dist}
222       }
223     }
224     if (maxdist == 0) { var nofibers = true; }
225     for (var i = 0; i < this.isotropicRegions.source.length; i++) {
226       var region = this.isotropicRegions.source[i];
227       var dist = Math.sqrt(
228         Math.pow(region.center[0],2) +
229         Math.pow(region.center[1],2) +
230         Math.pow(region.center[2],2)
231       ) + region.radius;
232       if (dist > maxdist) {maxdist = dist}
233     }
234     if (nofibers) {
235       if (maxdist > 0) {
236         maxdist *= 5;
237       } else {
238         maxdist = 1;
239       }
240     }
241     return maxdist;
242   },
243   newFiber: function() {
244     /** @function newFiber
245       * @memberof Phantom
246       * @desc Creates a new <i>blank</i> fiber in the scene.
247       <br>The fiber features two points in X axis with {@link Phantom.
              radius|phantom's radius} distance.
248       <br>Will {@link render}.
249       */
250
251     // New fiber will feature two points, following the x axis.
252     var cp = [
253       [-1 * roundToPrecision(this.radius()), 0, 0],
254       [roundToPrecision(this.radius()), 0, 0],
255     ];
256     var radius = roundToPrecision(this.radius() / 10);
257     // Add nbElements in parameters so this.addFiber calculates segments
          by itself
258     var parameters = {
259       nbElements: this.fibers.source.length + this.isotropicRegions.source
            .length
```

```
260     };
261
262     this.addFiber(new FiberSource(cp, 'symmetric', radius), parameters);
263   },
264   newIsotropicRegion: function() {
265     /** @function newIsotropicRegion
266       * @memberof Phantom
267       * @desc Creates a new <i>blank</i> isotropic region in the scene.
268       <br>The isotropic region will be centered in the scene and have 1/5
              of {@link Phantom.radius|phantom's radius} as radius.
269       <br>Will {@link render}.
270       */
271
272     // New region is to stay in the center. Radius is set to be a fifth of
          phantom radius.
273     var center = [0, 0, 0];
274     var radius = roundToPrecision(this.radius() / 5);
275     // Add nbElements in parameters so this.addIsotropicRegion calculates
          segments by itself
276     var parameters = {
277       nbElements: this.fibers.source.length + this.isotropicRegions.source
              .length
278     };
279
280     this.addIsotropicRegion(new IsotropicRegionSource(center, radius),
          parameters);
281   },
282   resetColors: function(){
283     /** @function resetColors
284       * @memberof Phantom
285       * @desc Resets color of all bundles. Important when unhighlighting
              with {@link Phantom#highlightColor|highlightColor} being defined
              .
286       <br>Will {@link render}.
287       */
288
289     // Color contained in their objects is given back to material's meshes
          .
290     for (var i = 0; i < this.fibers.tube.length; i++) {
291       this.fibers.tube[i].mesh.material.color = this.fibers.tube[i].color;
292     }
293     for (var i = 0; i < this.isotropicRegions.sphere.length; i++) {
294       this.isotropicRegions.sphere[i].mesh.material.color = this.
              isotropicRegions.sphere[i].color;
295     }
296     // Render so changes are made visible
297     render();
298   },
299   fadeAll: function(opacity) {
300     /** @function fadeAll
301       * @memberof Phantom
302       * @param {Number} [opacity=Phantom.highlightOpacity] Opacity to fade
              to (over 1)
303       * @desc Fades all bundles to given opacity.
```

```
304       <br>Will {@link render}.
305       */
306
307     // Only reset colors if custom highlightColor is enabled
308     if (this.highlightColor) this.resetColors();
309     // Set default opacity in case none is specified
310     if ((opacity === undefined) || (opacity > 1) || (opacity < 0)) {
311       opacity = this.highlightOpacity;
312     }
313     for (var i = 0; i < this.fibers.tube.length; i++) {
314       this.fibers.tube[i].mesh.material.opacity = opacity;
315     }
316     for (var i = 0; i < this.isotropicRegions.sphere.length; i++) {
317       this.isotropicRegions.sphere[i].mesh.material.opacity = opacity;
318     }
319     // Render so changes are made visible
320     render();
321   },
322   unfadeAll: function() {
323     /** @function unfadeAll
324       * @memberof Phantom
325       * @desc Unfades all bundles, setting opacity to 1.
326       <br>Will {@link render}.
327       */
328
329     // Only reset colors if custom highlightColor is enabled
330     if (this.highlightColor) this.resetColors();
331     // Opacity 1 is given back
332     for (var i = 0; i < this.fibers.tube.length; i++) {
333       this.fibers.tube[i].mesh.material.opacity = 1;
334       this.fibers.tube[i].mesh.renderOrder = 0;
335     }
336     for (var i = 0; i < this.isotropicRegions.sphere.length; i++) {
337       this.isotropicRegions.sphere[i].mesh.material.opacity = 1;
338       this.isotropicRegions.sphere[i].mesh.renderOrder = 0;
339     }
340     // Render so changes are made visible
341     render();
342   },
343   addToScene: function(scene) {
344     /** @function addToScene
345       * @memberof Phantom
346       * @param {THREE.Scene} scene Scene in which the Phantom will be
              added to.
347       * @desc Adds all Phantom bundles to given scene.
348       <br>Will {@link render}.
349       */
350
351     // Scene is cleared so that present meshes do not disturb
352     scene.removePhantom();
353     this.unfadeAll();
354     this.fibers.tube.forEach(function(tube) {
355       scene.add(tube.mesh)
356     });
```

```
357     this.isotropicRegions.sphere.forEach(function(sphere) {
358       scene.add(sphere.mesh)
359     });
360     // Render so changes are made visible
361     render();
362   },
363   addAsSkeleton: function(scene) {
364     /** @function addAsSkeleton
365       * @memberof Phantom
366       * @param {THREE.Scene} scene Scene in which the Phantom will be
              added to.
367       * @desc Adds all Phantom bundles to given scene in a Skeleton form..
368       <br>Will {@link render}.
369       */
370
371     // Phantom is added as faded tubes. Opacity 75% than default.
372     this.addToScene(scene);
373     this.fadeAll(this.highlightOpacity*.75);
374     // Skeleton structure added
375     this.fibers.skeleton.forEach(function(skeleton) {
376       scene.add(skeleton.line, skeleton.spheres)
377     });
378     // Render so changes are made visible
379     render();
380   },
381   fiberHighlight: function(n) {
382     /** @function fiberHighlight
383       * @memberof Phantom
384       * @param {Number} n Index of the fiber to highlight.
385       * @desc Fades all but the given fiber. Highlight opacity cannot be
              specified.
386       <br>Will {@link render}.
387       */
388
389     // Fade all but wanted fiber
390     this.fadeAll();
391     this.fibers.tube[n].mesh.material.opacity = 1;
392     this.fibers.tube[n].mesh.renderOrder = -1;
393     // If custom highlight color, apply.
394     if (this.highlightColor) {
395       this.fibers.tube[n].mesh.material.color = this.highlightColor;
396     }
397     // Render so changes are made visible
398     render();
399   },
400   regionHighlight: function(n) {
401     /** @function regionHighlight
402       * @memberof Phantom
403       * @param {Number} n Index of the region to highlight.
404       * @desc Fades all but the given region. Highlight opacity cannot be
              specified.
405       <br>Will {@link render}.
406       */
407
```

```
408     // Fade all but wanted region
409     this.fadeAll();
410     this.isotropicRegions.sphere[n].mesh.material.opacity = 1;
411     this.isotropicRegions.sphere[n].mesh.renderOrder = -1;
412     // If custom highlight color, apply.
413     if (this.highlightColor) {
414       this.fibers.tube[n].mesh.material.color = this.highlightColor;
415     }
416     // Render so changes are made visible
417     render();
418   },
419   cpHighlight: function(fiberindex, controlpointindex, mode) {
420     /** @function cpHighlight
421       * @memberof Phantom
422       * @desc Overlays a colored slightly bigger sphere over a control
              point. Used for
423       the user focusing in this element.
424       <br>Will {@link render}.
425       * @returns {THREE.MeshBasicMaterial} The highlight mesh
426       * @param {Number} fiberindex Index of the fiber containing the
              control point to highlight.
427       * @param {Number} controlpointindex Index of the control point to
              highlight.
428       * @param {String} mode Highlight mode:
429       <ul>
430       <li>'red': Red color. This is the only one not be removed by {@link
              THREE.Scene.removeCPHighlight} unless specified.
431       <li>'blue': Blue color.
432       <li>'green': Green color. This one is 90% the size to avoid bad
              rendering in clashes with the orthers.</ul>
433       */
434
435     scene.removeCPHighlight();
436     var fiber = phantom.fibers.source[fiberindex];
437     var cp = fiber.controlPoints[controlpointindex];
438
439     var geometry = new THREE.SphereGeometry(fiber.radius/4, 16, 16);
440     var surface = new THREE.MeshBasicMaterial();
441
442     var mesh = new THREE.Mesh(geometry, surface);
443     mesh.isHighlight = true;
444     mesh.position.set(cp[0], cp[1], cp[2]);
445
446     switch (mode) {
447       case 'blue':
448         mesh.material.color = new THREE.Color(0x0000FF)
449         // this will be later used for filtering which points are to be
              removed.
450         mesh.isBlueHighlight = true;
451         break;
452       case 'red':
453         scene.removeCPHighlight(true);
454         mesh.material.color = new THREE.Color(0xFF0000)
455         mesh.position.set(guiStatus.formerCP[0], guiStatus.formerCP[1],
```

```
            guiStatus.formerCP[2]);
456         break;
457       case 'green':
458         mesh.material.color = new THREE.Color(0x00FF00);
459         mesh.scale.x = 0.9;
460         mesh.scale.y = 0.9;
461         mesh.scale.z = 0.9;
462         // this will be later used for filtering which points are to be
                removed.
463         mesh.isBlueHighlight = true;
464         break;
465       default: // Should not happen!
466         console.error('Incorrect cpHighlight mode! Was set as ' + mode);
467         mesh.material.color = new THREE.Color(0xFFFFFF)
468     }
469
470     scene.add(mesh);
471     render();
472
473     return mesh;
474   },
475   revealSkeleton: function(scene, n) {
476   /** @function revealSkeleton
477     * @memberof Phantom
478     * @desc Adds Phantom to the scene and fades all by adding a Skeleton
            fiber to a
479     given fiber. This fiber's tube will feature twice the default opacity
            for making the user stay focus.
480     <br>Will {@link render}.
481     * @param {THREE.Scene} scene Scene in which the Phantom will be added
            to.
482     * @param {Number} n Index of the fiber to highlight.
483     */
484     this.addToScene(scene);
485     this.fadeAll();
486     // Focus fiber is faded more so that thread can be seen with any
            problem
487     this.fibers.tube[n].mesh.material.opacity = this.highlightOpacity*2;
488     this.fibers.tube[n].mesh.renderOrder = -1;
489     scene.add(this.fibers.skeleton[n].line, this.fibers.skeleton[n].
            spheres);
490     // Render so changes are made visible
491     render();
492   }
493 }
```

2.3.21 save.js

```
1 /**@overview Contains functions regarding the parse, export and download
      process of a Phantom.*/
2 // Parsable objects must contain only those parameters the loaders expect
      to find
3
4 // Returns a JSON string with the phantom in ParsableFiber and
```

```
    ParsableRegion classes
Phantom.prototype.export = function() {
/** @function export
  * @memberof Phantom
  * @desc Parses Fibers and Isotropic Regions in the Phantom and returns a parsed, indented string.
  <br>The JSON is fully compatible with Phantomas. Index in the array is used as name.
  <br>Information from fibers:
  <ul><li>Control points
  <li>Tangents
  <li>Radius
  <li>Color
  </ul>
  Information from Isotropic Regions:
  <ul><li>Center
  <li>Radius
  <li>Color
  </ul>
  * @returns {String} Parsed variable ready for pushing to download.
*/
  // PRIVATE CONSTRUCTORS
  ParsableFiber = function(control_points, tangents, radius, color) {
    this.control_points = control_points;
    this.tangents = tangents;
    this.radius = Number(radius);
    this.color = Number(color.getHex());
  }
  ParsableRegion = function(center, radius, color) {
    this.center = center;
    this.radius = Number(radius);
    this.color = Number(color.getHex());
  }


  var parsable_phantom = new Object;
  parsable_phantom.fiber_geometries = new Object;
  parsable_phantom.isotropic_regions = new Object;

  // FIBERS
  this.fibers.source.forEach( function(source, index) {
    var control_points = [];
    // Control Points are expected in a unique string.
    source.controlPoints.forEach( function(cp) {
      cp.forEach( function(element){
        control_points.push(element);
      });
    });

    var parsable_fiber = new ParsableFiber(control_points, source.tangents, source.radius, source.color);
    // Fiber names featured in Phantomas are not featured here. Instead, numbers are applied.
    parsable_phantom.fiber_geometries[index.toString()] = parsable_fiber;
```

```
54    });
55
56    // ISOTROPICREGIONS
57    this.isotropicRegions.source.forEach( function(source, index) {
58      var parsable_region = new ParsableRegion(source.center, source.radius,
            source.color);
59      // Fiber names featured in Phantomas are not featured here. Instead,
            numbers are applied.
60      parsable_phantom.isotropic_regions[index.toString()] = parsable_region
            ;
61    });
62
63    // null,2 automatically indents json file
64    var parsed_phantom = JSON.stringify(parsable_phantom, null, 2);
65
66    return parsed_phantom;
67  }
68
69  pushDownload = function(content) {
70  /** @function pushDownload
71    * @desc Pushes to the navigator the download of a string as a
          ddmmyyyyhhmm-phantom_save.JSON file.
72    <br> Requires an &lt;a&gt; element in the HTML with id="
          downloadAnchorElem".
73    * @param {string} content A string containing the content to be included
           in the file.
74  */
75
76    // From http://stackoverflow.com/questions/19721439/download-json-object
          -as-a-file-from-browser
77    function timestamp() {
78      // Partly from https://stackoverflow.com/questions/12409299/how-to-get
            -current-formatted-date-dd-mm-yyyy-in-javascript-and-append-it-to-
            an-i
79      var today = new Date();
80      var yyyy = today.getFullYear();
81      var mm = today.getMonth()+1; //January is 0!
82      if(mm<10){
83        mm='0'+mm;
84      }
85      var dd = today.getDate();
86      if(dd<10){
87        dd='0'+dd;
88      }
89      var hh = today.getHours();
90      if (hh<10) {
91        hh='0'+hh;
92      }
93      var mn = today.getMinutes();
94      if (mn<10) {
95        mn='0'+mn;
96      }
97
98      var timestamp = dd+mm+yyyy+hh+mn;
```

```
99      return timestamp;
100   }
101   var uriContent = "data:text/json;charset=utf-8," + encodeURIComponent(
          content);
102   var dlAnchorElem = document.getElementById('downloadAnchorElem');
103   dlAnchorElem.setAttribute("href", uriContent);
104   dlAnchorElem.setAttribute("download", timestamp()+"-phantom_save.json");
105   dlAnchorElem.click();
106 }
```

2.3.22 dragAndDrop.js

```
1  /** @overview Contains THREE.js functions responsible of the Drag and
       Dropping feature.
2  */
3
4  function dragAndDrop() {
5    /** @function dragAndDrop
6      * @desc Builds or resets Drag and Drop interactive controls in the
           scene.
7      */
8
9    var control = new THREE.TransformControls(camera, renderer.domElement);
10
11   scene.removeControls();
12   scene.removeCPHighlight();
13   var object = phantom.cpHighlight(guiStatus.editingFiber, guiStatus.
         editingCP, 'green');
14
15   control.name = 'dragAndDrop';
16
17   control.addEventListener('change', function() {
18     var pos = this.object.position;
19     pos.x = roundToPrecision(pos.x);
20     pos.y = roundToPrecision(pos.y);
21     pos.z = roundToPrecision(pos.z);
22
23     document.getElementById('xvalue').value = pos.x;
24     document.getElementById('yvalue').value = pos.y;
25     document.getElementById('zvalue').value = pos.z;
26     render();
27
28     document.getElementById('guiFiberLength').innerHTML = roundToPrecision
           (phantom.fibers.source[guiStatus.editingFiber].length);
29   });
30
31   control.addEventListener('mouseUp', function() {
32     var pos = object.position;
33     phantom.fibers.source[guiStatus.editingFiber].setControlPoint(
           guiStatus.editingCP, 'x', pos.x);
34     phantom.fibers.source[guiStatus.editingFiber].setControlPoint(
           guiStatus.editingCP, 'y', pos.y);
35     phantom.fibers.source[guiStatus.editingFiber].setControlPoint(
           guiStatus.editingCP, 'z', pos.z);
```

```
36    });
37
38
39    control.attach(object);
40    scene.add(control);
41    render();
42  }
43
44  THREE.Scene.prototype.removeControls = function() {
45  /** @method removeControls
46    * @memberof module:THREE.Scene
47    * @desc Removes all Drag and Drop controls present in the Scene.
48    */
49    var remove = [];
50    var scene = this;
51    scene.children.forEach(function(object) {
52      if (object.name == 'dragAndDrop') {
53        remove.push(object);
54      }
55    });
56    remove.forEach( function(object) {
57      object.dispose();
58      scene.remove(object);
59    });
60    render();
61  }
```