Javascript viewer
For NMR and MS experiment

Tutorial to know how it works

Samy Deghou
Elève ingénieur 5ème année
Majeure Bioinformatique

Contents

1   Introduction

1.1   What does it do?

The aim is to build a viewer displaying the output of a chemical experiment:

-Mass Spectra experiment

-NMR experiment

Each output experiment provides the information about:

-The molecule studied.

-Its spectra.

-The additional information about the condition of the experiment

Hence, the viewer must **displays** the **molecule** along with its **spectrum**. Besides, we want it to be **interactive** with the user, e.g when the user mouse over one peak, it has to **highlights** and highlights the corresponding atoms it is linked to(in the case of a NMR experiment) and it has to display the corresponding fragment(s) (in case of a MS experiment).

1.2   How do we do it?

The entire viewer is build with JavaScript and use the google closure library
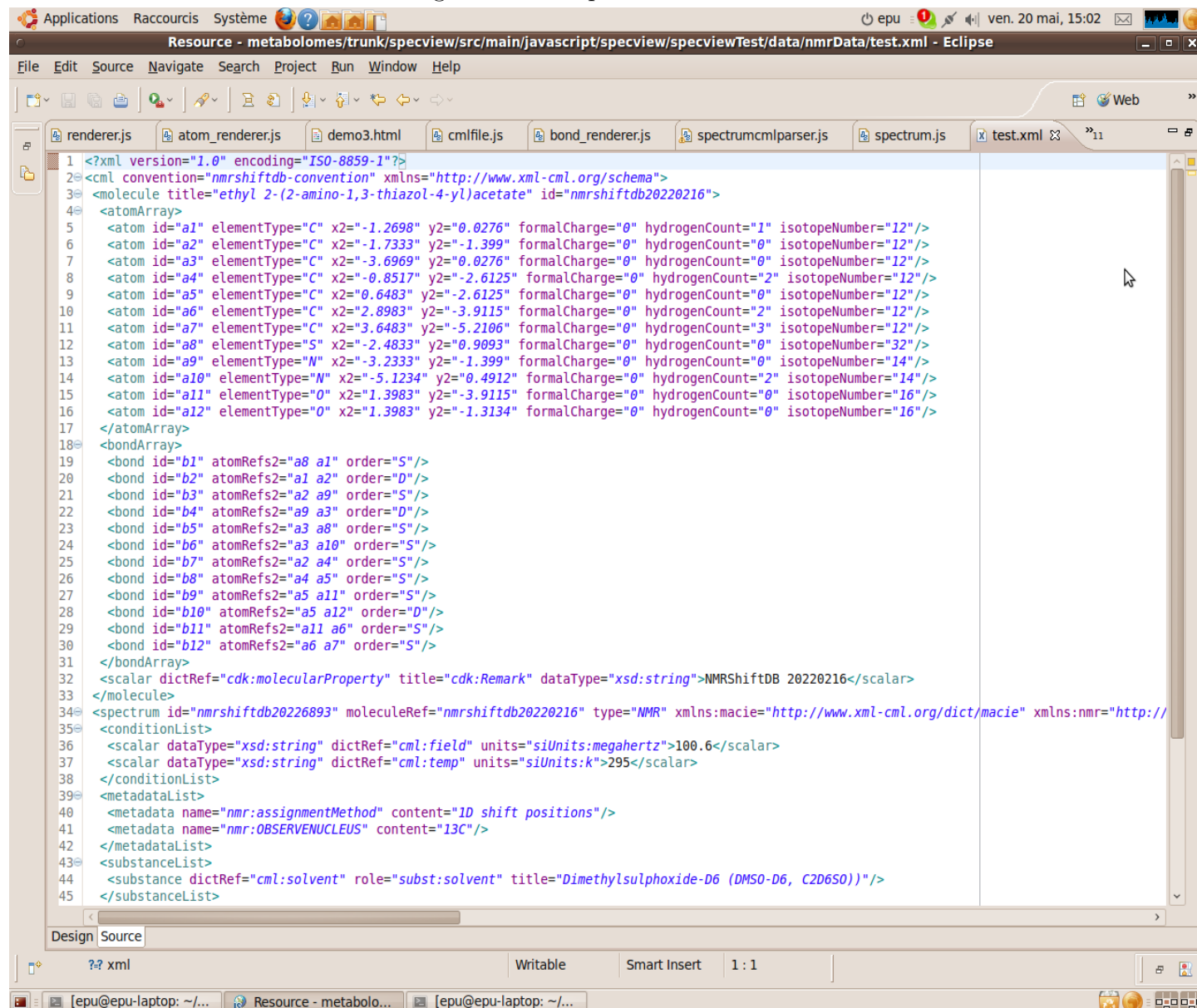
1.2.1   Google Closure

In summary, the Closure Library is a JavaScript library which is based on a modulable architecture. It provides cross-browser functions for DOM manipulations and events, AJAX and JSON, as well as more high-level objects such as User Interface widgets and controls. For instance and in our case, it provides the tools to design a canvas editor, and all the objects to have a control over this canvas.

2 The entry point

Our Input data is a file in a **cml file**, which is a standart of xml. The file contains the information generated by the output of the experiment. It holds either MS data, or NMR data.

Figure 1: Example of cml file

2.1 What does it contain?

This file provides all the information required in order to build a **molecule** and a **spectrum**. THe molecule consists in a liste of ***atoms***: The bonds refer to 2 atoms by their id. It is possible, that

Figure 2: A list of atoms

```
<atomArray>
 <atom id="a1" elementType="C" x2="-1.2698" y2="0.0276" formalCharge="0" hydrogenCount="1" isotopeNumber="12"/>
 <atom id="a2" elementType="C" x2="-1.7333" y2="-1.399" formalCharge="0" hydrogenCount="0" isotopeNumber="12"/>
```

The list of atoms contains all the informations to identify the entities of the molecules. The important point is that the coordinates it contains are relative coordinates. In order to display the atoms on the screen, they have to be transformed into actual coordinates, "pixel coordinates".

Figure 3: A list of bonds

```
<bondArray>
 <bond id="b1" atomRefs2="a8 a1" order="S"/>
 <bond id="b2" atomRefs2="a1 a2" order="D"/>
```

single order bond have a certain stereo specificity. In that case, the bond "b_i''" would have a tag "*bondStereo''*" specifying the orientation of the bond. Thie information about the **peak** os probably

Figure 4: Example of peak annotation for a NMR experiment

```
<peakList>
 <peak xValue="103.12" xUnits="units:ppm" peakShape="sharp" peakMultiplicity="D" id="p0" atomRefs="a1'
```

the most important. This where the **annotation** stands. Each peak has a **field** or a **child** that refer to:

-One or multiple **atom** if the file holds **NMR** experiment

-One or multiple **molecule** if the file holds **MS** experiment

The objects are mentioned by their inner identifier(identifier found in the file, e.g a_1,a_2... for an atom and m_1,m_2... for a molecule).

Furthermore, MS files can have fields tagged "scalar information" at the end of each molecule. These file holds additional information on the moleculem, such as physico chemical properties, inChi key, other identifiers...

Finally, the file contains other information regarding the experiment within the tag "condition" and "metadata".

3 Create a JavaScript Object

Now, in order to display the content of the file, we decide to create an object, called *"metaSpecObject"*. This object is a generic object that can contains all the data to represent a molecule along with its spectrum and the information to allow the cross talk between the two entities.

Figure 5: Attributes of the metaSpecObject

```
/**
 * THe experience type that the file holds, e.g MS or NMR.
 */
this.experienceType="";
/**
 * The main molecule of the experiment. In case of NMR, it would be the only molecule. In case of MS, it would
 * be the reactant molecule.
 */
this.molecule=null;
/**
 * The only spectrum of the experiment as found in the file
 */
this.spectrum=null;
/**
 * Only available with MS data. Array holding all the fragment molecule of the reactant molecule as found in the file
 */
this.secondaryMolecule=null;
/**
 * These arrays are useful when dealing with NMR data.IN NMR data, each peak is associated with at least one atom
 */
this.ArrayOfAtoms=new Array(); // Keys are the inner atom id. Values are the atom objects
this.ArrayOfBonds=new Array();
this.ArrayOfPeaks=new Array();
this.ArrayOfSecondaryMolecules=new Array();//{m1:molecule1;m2:moleucle2...}
/**
 * The box of the original fragment of the ms Experiment. Every other fragment has to be drawn in that box.
 */
this.mainMolBox=null;
/**
 * The box of the spectrum. Calculated according the the mainMolBox
 */
this.mainSpecBox=null;
/**
 * transform object.
 * When an atom is parsed from the file, it contains relative coordinates that usually and preferably are in the
 * range [-10;10]. In ordered to be displayed to the screen, these coordinates has to be transformed according to
 * the parameter of the controller object.
 * These coordinates are transformed in the method:
 *  {@see specview.model.nmrData.prototype.setCoordinatesWithPixelOfMolecule}.
 * There we create a transform object and use it to transform the coordinate,
 * It is however to keep track of that transform object that we creates, because it is still used in other classes,
 * for instance in {@see double_bond_renderer.js}. And in order to make sure that we use make the same
 * transformations (e.g use the same transform object) we shall make an attribute out of it of the object nmrData.
 */
this.transform=null;
```

The aim is to conceive an object that holds all the elements to create a molecule object, a spectrum object, and keeping track of the relation between one peak and another object(either a molecule or an atom). This is way, each peak is considered as being an object (@see peak.js) and has several fields that may relate to its corresponding objects:

Figure 6: The object Peak

```
/**
 * If the cml file holds NMR experiment data, and if it is an annotated file, then the peak are linked to one or
 * multiple atoms. atomref is an atom identifier or an array of atom identifier found in the file.
 */
this.atomRef=opt_atomRef;// Should be an array
|

/**
 * If the file holds Mass Spec experiment, and if the file is annotated, then some peaks are related to one or many
 * molecules.
 * arrayOfSecondary molecules contains the molecule identifiers that refers to the molecules.
 */
this.arrayOfSecondaryMolecules=opt_molRefs;//normally just one molecule
```

With these two fields, whenever a peak is linked to another object, it is possible to track this object.

4 The parser

So far the parser takes cml files (version 2008–¿2010) and works with Firefox and Internet Explorer. N.B: Javascript does not conveniently work with file input stream, so the most convenient way to deal with it is to:

(i) Make a string out of the cml file.

(ii) Load the cml string into an XML object.

(iii) Parse the object and extract the information we want to create the metaSpecObject

4.1 A few points

Before rendering everything, some things have to be done

4.1.1 SetCoordinatesWithPixel

The file contains relative coordinates, that is, they need to be transfomed into actual coordinates in order to be properly rendered. This is done in the method setCoordinateWithPixel: That method

Figure 7: The object Peak

```
/**
 * Set the pixel coordinate of the molecule
 * Create a box or the specturm according to the pixel coordinates of the molecule
 * Set the pixel coordinate of the spectrum according the the spectrum box that has been created
 * @param editorSpectrum
 * @param zoomX
 */
specview.model.NMRdata.prototype.setCoordinatesWithPixels = function(editorSpectrum){
    this.mainMolBox=this.getMoleculeBox(editorSpectrum);
    this.setCoordinatesPixelOfMolecule(editorSpectrum);
    this.mainSpecBox=this.getSpectrumBox();
    this.setCoordinatesPixelOfSpectrum();
};
```

is an instance method of the metaSpecObject. The first step is to define:

(i) Get the molecule box where the molecule will be drawn.

(ii) Set the pixel coordinates of the molecule.

(iii) Set the spectrum box according to the molecule box.

(iv) Set the pixel coordinates of the spectrum(e.g of all the peaks)

Get the molecule box

Figure 8: Get the molecule box

```
/**
 * Build a box for the molecule.
 * We first need to get a box of relative coordinates out of the relative coordinates of the molecule(coordinates found
 * in the file). That is Step 1.
 * Then, we need to transform these coordinates into pixel coordinates in order to render them. To do so, we need the
 * object transform. That is Step 2
 *
 */
specview.model.NMRdata.prototype.getMoleculeBox = function(editorSpectrum){
    /*
     * Step 1
     */
    var molecule=this.molecule;
    var box=molecule.getBoundingBox();
    var boxTopLeftCoord =new goog.math.Coordinate(box.left,box.top);
    var boxTopRightCoord =new goog.math.Coordinate(box.right,box.top);
    var boxBotLeftCoord =new goog.math.Coordinate(box.left,box.bottom);
    var boxBotRightCoord =new goog.math.Coordinate(box.right,box.bottom);
    boxTopRightCoord=(boxTopRightCoord.x<boxTopLeftCoord.x ? new goog.math.Coordinate(1200,box.top) : boxTopRightCoord);
    boxBotRightCoord=(boxBotRightCoord.x<boxBotLeftCoord.x ? new goog.math.Coordinate(1200,box.bottom) : boxBotRightCoord);
    /*
     * Step 2
     */
    var atom_coords=goog.array.map(this.molecule.atoms,function(a) {return a.coord; });//the coords of the file. Simple array
    var relative_box=goog.math.Box.boundingBox.apply(null, atom_coords);
    var scaleFactor = 0.90;
    var widthScaleLimitation = 0.4;
    var margin = 0.3;
    var editor=editorSpectrum;
    var ex_box=relative_box.expand(margin,margin,margin,margin);
    var transform = specview.graphics.AffineTransform.buildTransform(ex_box,widthScaleLimitation,editorSpectrum.graphics,scaleFactor);
    return transform.transformCoords( [boxTopLeftCoord,boxTopRightCoord,boxBotLeftCoord,boxBotRightCoord]);
};
```

The method take as an input the editor object, and return an array of 4 coordinates (pixel coordinates) that will be used to draw the molecule box.

Transform object

In order to transform relative coordinates into actual coordinates, we need to take into account the dimension(e.g width andheight) of the canvas editor (which is an attribute of the editor object). This is the reason why the method has an input "editorSpectrum" which is the editor object. This is used in the method to create a "transform" object which does the transformation from realtive coordinates into actual coordinates.

Set the molecule coordinates

Figure 9: Set the molecule Coordinates

```
/**
 * Set the pixel coordinates of the molecule.
 * Use the relative coordinate of the molecule found in the file.
 * Use the transform object of the google closure library. This object is an attribute of the editor(the controller)
 * and is dependant on the canvas created. Hence, this function must have an editor as an argument to extract the transform object
 * that will transform the relative coordinates into pixel coordinates.
 * @param editorSpectrum
 * @param zoomX
 */
specview.model.NMRdata.prototype.setCoordinatesPixelOfMolecule = function(editorSpectrum){
    var molecule=this.molecule; var spectrum=this.spectrum; var editor=editorSpectrum;
    var molBox = molecule.getBoundingBox();//CREATE THE MOLECULE BOX. THIS WILL ALLOW TO SET THE PARAMETER FOR EVERY OTHER OBJECTS
    var molHeight=Math.abs(molBox.top-molBox.bottom);
    var molWidth=Math.abs(molBox.left-molBox.right);
    var size=Math.max(molHeight,molWidth);
    var top=molBox.bottom;
    var bottom=top-size;
    var left= 1.1*molBox.right;
    var right=left+size;
    var bottom=-5.5;
    var right=23.5;
    var boxxx=new goog.math.Box(top,right,bottom,left);//CREATE THE SPECTRUM BOX ACCORDING TO THE MOLECULE BOX
    var atom_coords = goog.array.map(molecule.atoms,function(a) {return a.coord; });//the coords of the file. Simple array
    var peak_coords = goog.array.map(spectrum.peakList,function(a) {return a.coord;});
    var box = goog.math.Box.boundingBox.apply(null, atom_coords);
    var margin = 0.3;//this.config.get("margin");
    var ex_box = box.expand(margin, margin, margin, margin);
    var scaleFactor = 0.90;
    var widthScaleLimitation = 0.4;
    var trans = specview.graphics.AffineTransform.buildTransform(ex_box, widthScaleLimitation, editor.graphics, scaleFactor);
    this.transform=trans;
        goog.array.forEach(molecule.atoms,
          function(atom){
            var point = trans.transformCoords([ atom.coord ])[0];//point is the coordinates with pixelS
            atom.setPixelCoordinates(point.x, point.y);
        //   specview.model.NMRdata.logger.info(point.x+"  ;  "+point.y);
        });
};
```

Get the spectrum box Set the spectrum pixel coordinates

Figure 10: Get the spectrum box

```
/**
 * Build a box for the spectra out of the max coordinates of the molecule.
 * @param downLimit,rightLimit,upperLimit
 * These constant parameter are the limit of the canvas. They should be more flexible. To do so
 * we should pass the editor as an argument and extract its width and height.
 */
specview.model.NMRdata.prototype.getSpectrumBox = function(){
    var molecule=this.molecule;
    var maxY=0;
    var maxX=0;
    var downLimit=280;var rightLimit=1100;var upperLimit=5;
    goog.array.forEach(molecule.atoms,
        function(atom){
            if(atom.xPixel>maxX){maxX=atom.xPixel;}
            if(atom.yPixel>maxY){maxY=atom.yPixel;}
        });
    var box = new goog.math.Box(upperLimit,rightLimit,downLimit,maxX+40);
    var topLeftBox=new goog.math.Coordinate(box.left,box.top);
    var topRightBox=new goog.math.Coordinate(box.right,box.top);
    var bottomRightBox=new goog.math.Coordinate(box.right,box.bottom);
    var bottomLeftBox=new goog.math.Coordinate(box.left,box.bottom);
    return new Array(topLeftBox,topRightBox,bottomLeftBox,bottomRightBox);
};
```

Figure 11: Set the pixel coordinates pixel

```
/**
 * The spectrum coordinates(coordinates of the peaks) are simply calculated on the basis of its spectra box.
 * @param zoomX
 */
specview.model.NMRdata.prototype.setCoordinatesPixelOfSpectrum = function(){
    var spectrum=this.spectrum;
    var minX=spectrum.getMinValue();
    var maxX=spectrum.getMaxValue();
    var maxHeightOfPeak=spectrum.getMaxHeightPeak(); var maxValueOfPeak;  var adjustXvalue; var adjustYvalue;
    var sortedArray=spectrum.sortXvalues();
    var arrayOfPeakSorted=spectrum.mapPeakToxValue(sortedArray);
        maxValueOfPeak=spectrum.getMaxValuePeak();
        var bottomBoxLimit;
        var upperBoxLimit;
        var ecart=this.mainSpecBox[1].x-this.mainSpecBox[0].x;
        var valueToAdd=this.mainSpecBox[0].x;
        goog.array.forEach(spectrum.peakList,
            function(peak) {
            if(peak.intensity==maxHeightOfPeak){
                adjustYvalue=20;
                upperBoxLimit=adjustYvalue-10;
            }else{
                adjustYvalue=20/(peak.intensity/maxHeightOfPeak);
            }
            if(peak.xValue==maxValueOfPeak){
                adjustXvalue=ecart-4;
            }else{
                adjustXvalue=(peak.xValue*(ecart-4))/maxValueOfPeak;
            }
            var whereAllThePeakStartFrom=280;
            peak.isVisible=(adjustXvalue+valueToAdd<this.mainSpecBox[1].x &&
                           adjustXvalue+valueToAdd>this.mainSpecBox[0].x) ? true : false;
            peak.setCoordinates(adjustXvalue+valueToAdd,whereAllThePeakStartFrom,adjustXvalue+valueToAdd,adjustYvalue);
        },
        this);
        spectrum.setExtremePixelValues();
        bottomBoxLimit=280;
```

4.1.2 Mapping

The viewer has to be interactive. When the user mouse over peaks or atom or bonds, it has to highlights all the objects in relation with the focused object. To do so, we have to create a "neighborlist" object.

The "neighborlist" object

The controller object contains one attribute "neighborlist". It is a class for locating the objects nearest to a specified coordinate. That way, when the mouse will hover an object, the controller will know that at this precise coordinate is located an object that has to be highlighted. When we create a "neighborlist" object, we create an associative array whose keys are coordinates and value are the object at the given coordinate.

5 How to display everything?

Right after the "neighborlist" object has been created, we can render the whole thing.

Figure 12: THe instance method "render" of the controller object

```
specview.controller.Controller.prototype.render = function() {
    goog.array.forEach(this.models, function(model) {

        if (model instanceof specview.model.NMRdata) {
            var molecule=model.molecule;
            var spectrum=model.spectrum;
            var molBox=model.mainMolBox;
            var specBox=model.mainSpecBox;
//              this.spectrumRenderer.setBoundsBasedOnMolecule(molecule);
            atom_coords = goog.array.map(molecule.atoms,function(a) {return a.coord; });//the coords of the file. Simple ar
            peak_coords = goog.array.map(spectrum.peakList,function(a) {return a.coord;});
            box = goog.math.Box.boundingBox.apply(null, atom_coords);
            if(model.experienceType=="ms"){
                box.top=box.top-box.top;
                box.bottom=box.bottom-box.top;
                box.right=box.right-box.top;
                box.left=box.left-box.top;
            }
            margin = 0.3;//this.config.get("margin");
            ex_box = box.expand(margin, margin, margin, margin);
            scaleFactor = 0.90;
            widthScaleLimitation = 0.4;
            trans = specview.graphics.AffineTransform.buildTransform(ex_box, widthScaleLimitation, this.graphics, scaleFact
            this.moleculeRenderer.render(molecule,model.transform,molBox);
            this.spectrumRenderer.render(model,model.transform,specBox);
        }
    }, this);
};
```

At the end of this method are called this.**moleculeRenderer**.render(molecule,model.transform,molB
and this.**spectrumRenderer**.render(model,model.transform,specBox). "molceuleRenderer" and
"spectrumRenderer" are attributes of the controller object. More precisely they are classes to render
molecule and spectrum objects on graphics objects. These classes contain a instance method named
"render" which properly draw the molecule(respectibely spectrum) object on the graphics object.

Example of a renderer class

Figure 13: The instance method render of the class spectrumRenderer

```
/**
 * The spectrum is simply the object
 * Transform is static and has been set up in specview.controller.Controller.prototype.render.
 */
specview.view.SpectrumRenderer.prototype.render = function(metaSpecObject, transform, opt_box) {
    var spectrum=metaSpecObject.spectrum;
    this.setTransform(transform);
    var peakPath = new goog.graphics.Path();
    var peakStroke = new goog.graphics.Stroke(1.05,'black');
    var peakFill = null;
    //Draw the peaks
    goog.array.forEach(spectrum.peakList,
    function(peak) {
        if(peak.isVisible){
            peakPath.moveTo(peak.xPixel, peak.yPixel);
            peakPath.lineTo(peak.xTpixel,peak.yTpixel);
        }
    },
    this);
    this.graphics.drawPath(peakPath, peakStroke, peakFill);
```

At each "**this.graphics.drawPath(peakPath, peakStroke, peakFill)**" a new peak is drawn on the canvas editor.

6   Simple workflow

Figure 14: Step to render the experiment
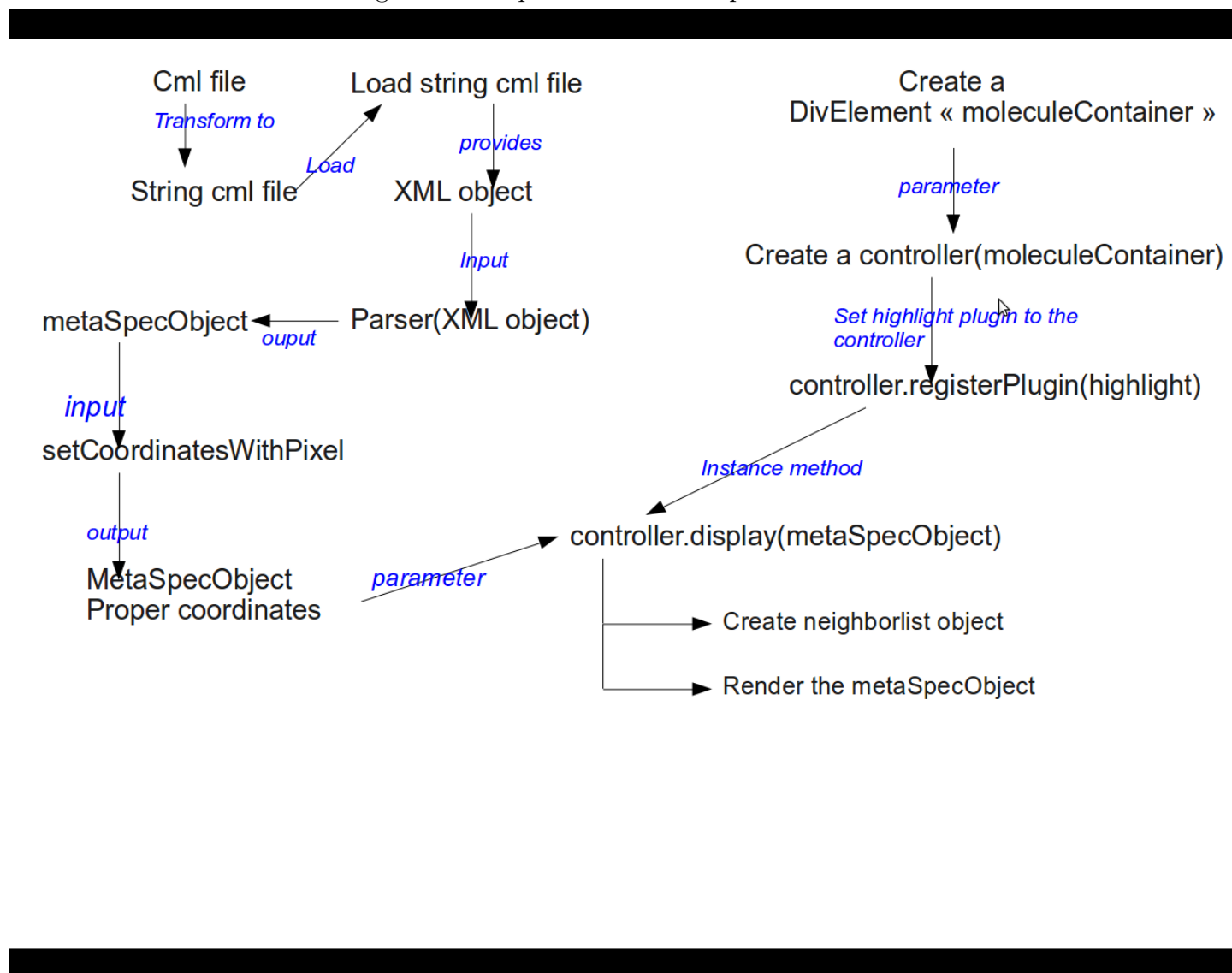
7 Highlight

The highlight process is a plugin that can be registered into the controller object.