Sockeye How-to: Adding or Editing New Sockeye Features

Authors: Stephen Montgomery / Gordon Robertson

Date: Jan 15th, 2004

IS THIS DOCUMENT FOR YOU:

- You have a new type of feature that you want to display in Sockeye
- The default organization of Sockeye features does not appeal to you

OUTLINE:

- This document will show a user how to edit or add new Sockeye features (A feature is a 3D object that represents an element of annotation in the Sockeye universe. For instance, by default, a gene feature is an orange cyclinder and an exon feature is a green cylinder).

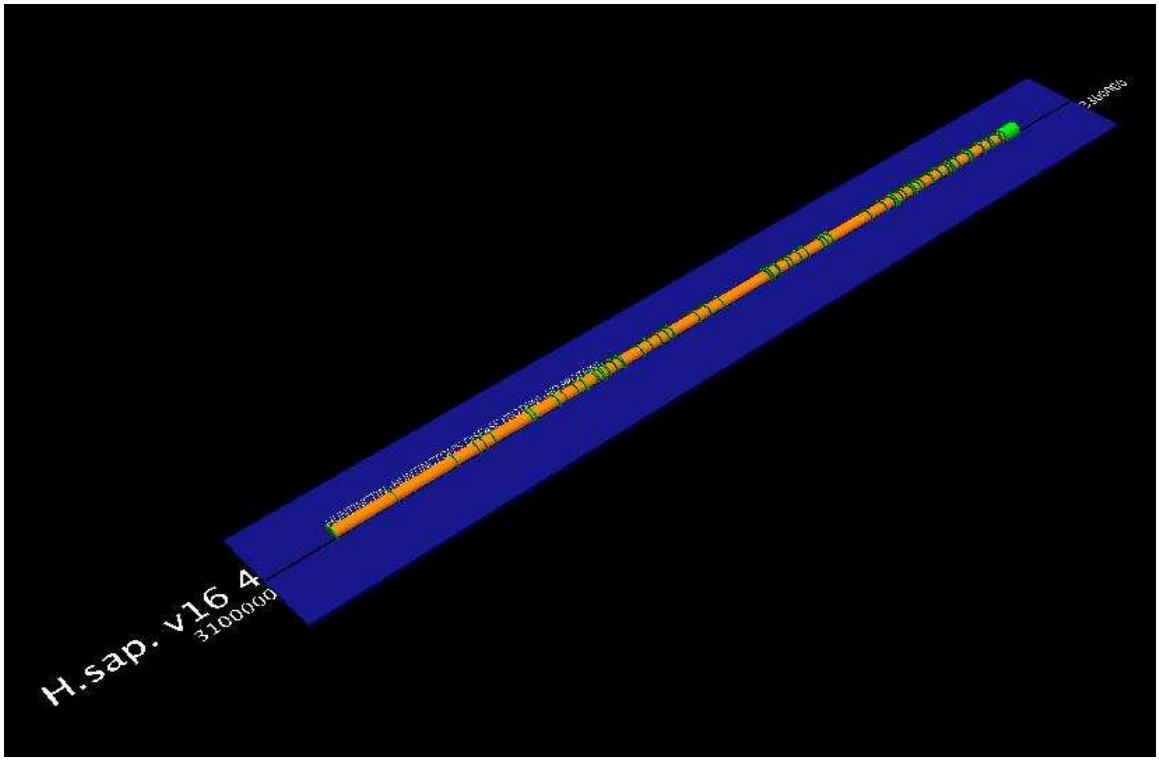

Figure 1: The huntington disease protein in Sockeye. An orange cyclinder represents a gene and the green cyclinders demarcate exons. Only the part of chromosome 4 that contains the HD-1 protein is being shown on this Sockeye track (blue plane).

HOW TO ADD OR EDIT A FEATURE:

1. All the features (3D objects) that are displayed in Sockeye are loaded from VRML files. (VRML: Virtual Reality Modelling Language). Unfortunately, VRML comes in more than one flavor and it is so powerful that it can define an entire 3D environment. While this is useful in some contexts, in Sockeye it would be confusing to have the environment change everytime someone loaded a new feature. To remedy this we only use very simple VRML files that describe 3D objects and that do not manipulate

the environment in Sockeye.

TO VIEW YOUR 3D MODEL FILES:

1. Go to your Sockeye directory
2. Go to `/resources/models/`
3. Here you will see all the 3D models that are used to make the default features in Sockeye. They all have a WRL file extension. They are text-based so if you feel like opening them, you can.

---

2. <u>All of the features that Sockeye displays are dynamically loaded from a configuration file.</u> This allows incredible flexibility for customizing Sockeye to your particular data.

TO VIEW YOUR FEATURE CONFIGURATION FILE:

1. Go to your Sockeye directory
2. Open your `user_config.xml` file
3. You are looking at an XML document that configures your Sockeye application.

---

3. Somewhere in your `user_config.xml` file you will encounter a group of XML tags called `<feature>`. These tags are responsible for defining the 3D properties of a given class of feature and their relationship within the feature tree. Without these tags, there would be nothing displayed in Sockeye's 3D environment or on the feature tree.

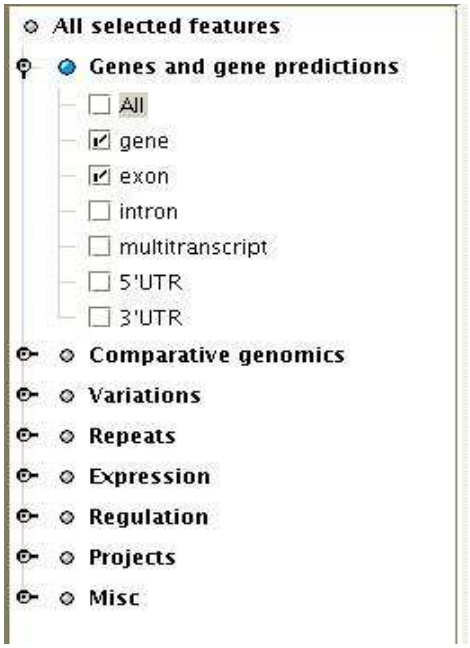

Figure 2: The Feature Tree

AN EXAMPLE OF A <feature>TAG

```
<feature>
          <name>exon</name>
          <parentName>gene</parentName>
          <display>
                    <color>0 170 0</color>
                    <hcolor>140 200 160</hcolor>
                    <shape>cylinder.WRL</shape>
                    <transparency>0</transparency>
                    <zoffset>0</zoffset>
                    <halfheight>false</halfheight>
                    <yoffset>0.1</yoffset>
                    <strandoffset>true</strandoffset>
                    <width>0.3</width>
                    <thickness>0.3</thickness>
                    <scoredthickness>false</scoredthickness>
                    <orientation>1</orientation>
                    <scaleable>true</scaleable>
                    <length>0.</length>
                    <alignment>center</alignment>
                    <collision>true</collision>
          </display>
</feature>
```

4. What does each tag mean?

1. name
   type: String
   E.g. gene, exon, Nucleocapsid protein

2. parentName
   type: String
   Set to another feature name to have the current feature displayed as a 'child' node under the 'parentName' node. Used to build the Feature Tree. E.g. **exon** under **gene**, **Nucleocapsid protein** under **gff_features**.
   IMPORTANT NOTE: A child feature inherits all the properties of the parent feature. So you are only required to specify the tags which have changed between the parent and the child.

   Feature display parameters

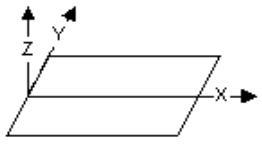

Figure 3: The Sockeye coordinate system

3.color
type: integer [space] integer [space] integer
Set an RGB colour by giving three integers between 0 and 255.

4.hcolor
type: integer [space] integer [space] integer
Set an RGB **highlight** colour by giving three integers between 0 and 255.
Only defined for exons. Used for persistent highlighting of exons for alternative transcripts. All other features use an internally defined transient highlight colour that does not need to be defined in the XML file.

5.shape
type: String
E.g. cylinder.WRL, cube.WRL, cone.WRL,...
Choose a VRML shape file from Sockeye's `/resources/models` directory.
You **may** be able to use VRML files that were not supplied with Sockeye, but such files may be incompatible and so are **not** supported.

6.transparency
type: decimal number
Set a number between 0.0 (opaque) and 1.0 (completely transparent).

7.halfheight
type: true / false
Set to **true** to make a feature stand **on** the 3D platform.
Typically such a feature will have a **score**, e.g. a degree of similarity between sequence regions, or an expression level.
Some **un**scored features may also stand on the platform.
Set to **false** if the feature should use `zoffset`.

8.zoffset
type: decimal number
fraction of the display track width.
The positive Z axis is shown in the coordinate system above.
Set the height of the **centre** of the feature **above** the surface of the platform.
Ignored if `halfheight` is **true**.

9.strandoffset
type: true / false
Set to **true** if the feature's data will include a strand indicator (e.g. **+** or **-**) that will be used to offset the 3D feature **to its strand's side** of the display track centreline.

Set the actual offset distance by `yoffset.`
A feature can be offset outside of the genomic strands by a larger offset.
If a feature is given a nonzero yoffset but `strandoffset` is **false**, its offset will be fixed, and will not depend on strand (e.g. Homologue).

`10.yoffset`
type: decimal number
fraction of the display track half-width
The positive Y axis is shown in the coordinate system above.
Set the distance across the platform that the feature' s centre is offset from the display track centre.
Can be positive, zero or negative, but negative cases will probably want `strandoffset` to be **true.**

`11.width`
type: decimal number
fraction of the display track half-width
Set the displayed width of the 3D feature.

`12.thickness`
type: decimal number
fraction of the display track half-width
Set the displayed height of the 3D feature.
To give a feature a symmetric cross-section (e.g. cylinder, sphere, cone), the width and thickness should be equal.

`13.scoredthickness`
type: true (default) / false
Set to **true** if the feature' s thickness should be scaled when it has a score.
Set to **false** if a feature should **not** be scaled **even if** it has a score (say, a ' gene' in a GFF file).

`14.orientation`
type: integer
Controls how the axes of a VRML shape are oriented relative to the axes of the platform, e.g. whether a VRML shape will be displayed with its long axis parallel to the platform' s X- or the Y-axis.
When `strandoffset=true` and `orientation=1`, a feature will have opposite display orientations on opposite strands.

`15.scaleable`
type: true / false
Set to **true** if the feature should be scaled by its genomic **start** and **end** coordinates (e.g. gene, exon, transcript,...), and **length** and **alignment** should be ignored .

Set to **false** if the feature should be displayed with a constant **length** and a controlled **alignment**.

16. `length`
    type: decimal
    fraction of display track width.
    If `scaleable` is **false**, set the (constant) display length for the feature (e.g.forward_primer).
    Ignored if `scaleable` is **true** (e.g. Gene).

17. `alignment`
    type: left / centre / right
    If `scaleable` is **false**, set how the feature should be located relative to the genomic region specified by its **start** and **end** coordinates (e.g. forward_primer).
    centre - centre of 3D feature = centre of Start/End region.
    left - for a feature on the + strand, the 5' end of the 3D feature will be at the Start coordinate; for the - strand, the 3' end will be at the End coordinate
    right - for a feature on the + strand, the 3' end of the 3D feature will be at the End coordinate; for the - strand, the 5' end will be at the Start coordinate
    Ignored if `scaleable` is **true** (e.g. Gene).

18. Collision
    type: true/false
    Allows for Java3D-based collision detection of features. If two features are colliding, one is moved vertically upwards. Experimental as Java3D doesn' t work well in multiple collision situations.

5. For a template see the feature named `default` in your `user_config.xml` file

---

WHAT THIS MEANS FOR YOUR GFF FILES

1. If you have been using GFF files to import your data into Sockeye. It probably hasn' t been very descriptive about what feature is what. By describing your features using different colors and shapes, you can very simply create a rich viewing environment for your information.

EXAMPLE:
Sample GFF Data

```
SARS-Associated BCGSC   Replicase 1A    250     13398   100     +       .
SARS-Associated BCGSC   Replicase 1B    13395   21467   100     +       .
SARS-Associated BCGSC   S (Spike) Glycoprotein  21477   25244   100     +       .
```

```
SARS-Associated BCGSC   E (SM) protein  26102   26332   100     +       .
```

Define <feature> tags for Replicase 1A, Replicase 1B, S (Spike) Glycoprotein, etc.

```
    <feature>
      <name>spike glycoprotein</name>
      <parentName>SARS annotation</parentName>
      <color>140 23 215</color>
      <Yoffset>0.3</Yoffset>
      <text>default_20</text>
   </feature>

    <feature>
      <name>replicase 1AB</name>
      <parentName>SARS annotation</parentName>
      <color>120 120 120</color>
      <Yoffset>0.9</Yoffset>
      <text>default_20</text>
   </feature>
```

Create detailed pictures

Figure 5: Sockeye picture of SARS virus on cover of Linux Journal

---

For more information on this topic contact sockeye@bcgsc.bc.ca.