Listing 52: VIZ Thickness Declaration

```
<viz:thickness value="2.0375757"/>
```

Node Shape Default node is shaped as a disc. Four shapes are proposed: *disc*, *square*, *triangle* and *diamond*. Images require an additional xml-attribute to set their location: uri.

Listing 53: Image Declaration

```
<viz:shape value="image" uri="http://my.image.us/blah.jpg"/>
```

Edge Shape Default edge is shaped as solid. Four shapes are proposed: *solid*, *dotted*, *dashed* and *double*.

6.1.2 Scale

The nodes/edges viz attributes described in previous section are stored as static data. This means that they don't mention why the viz:color has been set to this particular value. It's encoded as if the color has been chosen by hand without any data logic. But very often those viz attributes are actually computed from data attributes using for instance the Partition/Ranking features in Gephi. The scale features are meant to be able to inform Gexf consumers about the viz attributes logic in order to support legend drawing aside visualizing the graph.

Scales can be described at the graph level in the attributes element. Plural version of viz attributes: 'viz:colors', 'viz:sizes', 'viz:shapes', 'viz:positions', 'viz:thicknesses' are used to host the dynamic definitions of viz attributes.

We propose to include those into the the attribute(s) definitions in order to:

- Set the rule globally for the graph
- Reuse the node/edge class attribute already set in the attributes
- Nest the scale definition inside the corresponding attribute definition

To represent the dynamic definitions of viz attributes we need to define:

- which attribute to use: nested in attribute definition...
- the scale type: quantitative, qualitative, layout...
- the parameters: color hashmap, range, spline method...

A GEXF can host more than one for one viz parameter. For instance colors could be derived from two different node attributes type and degree. Therefore a *defaultscales* element can be used to specify which scale has been used to render the colors, sizes (...) in the gexf.

The scale information are meant to be used to draw a legend. But they can also theoretically be used to generate the according viz parameters. Yet a GEXF file can contain both viz attributes at the node/edge levels (local) and

a viz scale at the graph level (global). The local viz attributes must take over global definition when rendering visually a GEXF. To ensure the best GEXF compatibility among existing consumers we recommend to set both local and global systematically.

Scale Example The following gexf contains an gexf attributes section which embed colors, sizes and positions legend information.

Listing 54: VIZ Scales

```
<gexf xmlns="http://gexf.net/1.3.1"
      xmlns:viz="http://gexf.net/1.3.1/viz">
...
  <attributes class="node" mode="static">
    <attribute id="type" title="type" type="string">
      <viz:colors scale="qualitative">
        <viz:color forvalue="person" hex="#FF0000"/>
        <viz:color forvalue="city" hex="#00FF00"/>
      </viz:colors>
    </attribute>
    <attribute id="degree" title="Degree" type="integer">
      <default>0</default>
      <viz:sizes scale="quantitative" scalelabel="square-root">
        <viz:scalepoint forratio="0" factor="0" />
        <viz:scalepoint forratio="0.1" factor="0.316227766"/>
        <viz:scalepoint forratio="0.2" factor="0.447213595"/>
        <viz:scalepoint forratio="0.3" factor="0.547722558"/>
        <viz:scalepoint forratio="0.4" factor="0.632455532"/>
        <viz:scalepoint forratio="0.5" factor="0.707106781"/>
        <viz:scalepoint forratio="0.6" factor="0.774596669"/>
        <viz:scalepoint forratio="0.7" factor="0.836660027"/>
        <viz:scalepoint forratio="0.8" factor="0.894427191"/>
        <viz:scalepoint forratio="0.9" factor="0.948683298"/>
        <viz:scalepoint forratio="1.0" factor="1"/>
        <viz:range min="1" max="10" default="1" />
      </viz:sizes>
    </attribute>
    <viz:defaultscales colors="type" sizes="degree" />
    <viz:positions layoutalgorithm="forceatlas2"
      referenceURL="https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098679">
      <viz:param name="scale" type="integer" value="10"/>
      <viz:param name="stronger gravity" type="boolean" value="true"/>
    </viz:positions>
  </attributes>
  <attributes class="edge" mode="static">
    <attribute id="type" title="type" type="string">
      <viz:colors scale="qualitative">
        <viz:color forvalue="lives in" hex="#112345"/>
        <viz:color forvalue="works with" hex="#654523"/>
      </viz:colors>
    </attribute>
    <viz:defaultscales colors="type"/>
  </attributes>
...
</gexf>
```

Qualitative color scale The *viz:colors* describe a *qualitative* scale which uses the *modularityclass* node attribute to define the node's *viz:color* attribute. To do so the *viz:colors* element contains a color palette attributing a color for each value. A *viz:colordefault* can be used to handle null or not listed values.

Listing 55: Qualitative color scale

```
<attributes class="node" mode="static">
    <attribute id="modularityclass" title="Modularity Class" type="integer">
```

```
        <default>0</default>
        <viz:colors scale="qualitative">
            <viz:color forvalue="1" r="168" g="168" b="29"/>
            <viz:color forvalue="2" r="204" g="16" b="210"/>
            <viz:color forvalue="3" r="156" g="54" b="54"/>
            <viz:color forvalue="4" r="200" g="180" b="92"/>
            <viz:colordefault r="1" g="1" b="1"/>
        </viz:colors>
    </attribute>
</attributes>
```

Quantitative color scale The *viz:colors* can also describe a *quantitative* scale which uses the *degree* node attribute to define the node's *viz:color* attribute. To do so the *viz:colors* element contains a color gradient definition defining color points on a 0-1 value scale. The *scalelabel* attribute is useful to give more textual description of the scale logic.

Listing 56: Quantitative color scale

```
  <attributes class="node" mode="static">
  <attribute id="degree" title="Degree" type="integer">
      <default>0</default>
      <viz:colors scale="quantitative" scalelabel="greenish gradient">
          <viz:color forratio="0" hex="#EDF8FB"/>
          <viz:color forratio="0.5" hex="#66C2A4"/>
          <viz:color forratio="1" r="0" g="109" b="44"/>
      </viz:colors>
  </attribute>
</attributes>
```

Concurrent color scales There can be more than one *viz:colors* in the node attributes. To inform which scale were used to compute the colors encoded in the GEXF, a *viz:defaultscales* element nested inside *attributes* indicates which is the default scale to be used.

Listing 57: Quantitative color scale

```
  <attributes class="node" mode="static">
    <attribute id="modularityclass" title="Modularity Class" type="integer">
      <default>0</default>
      <viz:colors scale="qualitative">
          <viz:color forvalue="1" r="168" g="168" b="29"/>
          <viz:color forvalue="2" r="204" g="16" b="210"/>
          <viz:color forvalue="3" r="156" g="54" b="54"/>
          <viz:color forvalue="4" r="200" g="180" b="92"/>
          <viz:colordefault r="1" g="1" b="1"/>
      </viz:colors>
    </attribute>
    <attribute id="degree" title="Degree" type="integer">
        <default>0</default>
        <viz:colors scale="quantitative" scalelabel="greenish gradient">
            <viz:color forratio="0" hex="#EDF8FB"/>
            <viz:color forratio="0.5" hex="#66C2A4"/>
            <viz:color forratio="1" r="0" g="109" b="44"/>
        </viz:colors>
    </attribute>
    <viz:defaultscales colors="type" />
</attributes>
```

Quantitative size scale For *quantitative* scale, *viz:scalepoint* elements can be used to define an transformation function to adapt node attributes values to the viz (size here) parameters. The function is expressed as a set of points which represents a [0-1] to [0-1] curve. Here the *scalelabel* indicates that this set of points actually represent a square-root function. Although a set of discreet point is not as precise as a real function expression it has been preferred as function expression would have mean choosing a syntax and is more flexible than a finite set of function name.

Listing 58: Quantitative size scale

```
<attributes class="node" mode="static">
    <attribute id="degree" title="Degree" type="integer">
        <default>0</default>
        <viz:sizes scale="quantitative" scalelabel="square-root">
            <viz:scalepoint forratio="0" factor="0" ></viz:scalepoint>
            <viz:scalepoint forratio="0.1" factor="0.316227766" ></viz:scalepoint>
            <viz:scalepoint forratio="0.2" factor="0.447213595"></viz:scalepoint>
            <viz:scalepoint forratio="0.3" factor="0.547722558"></viz:scalepoint>
            <viz:scalepoint forratio="0.4" factor="0.632455532"></viz:scalepoint>
            <viz:scalepoint forratio="0.5" factor="0.707106781"></viz:scalepoint>
            <viz:scalepoint forratio="0.6" factor="0.774596669"></viz:scalepoint>
            <viz:scalepoint forratio="0.7" factor="0.836660027"></viz:scalepoint>
            <viz:scalepoint forratio="0.8" factor="0.894427191"></viz:scalepoint>
            <viz:scalepoint forratio="0.9" factor="0.948683298"></viz:scalepoint>
            <viz:scalepoint forratio="1" factor="1"></viz:scalepoint>
            <viz:range min="1" max="10" default="1" />
        </viz:sizes>
    </attribute>
</attributes>
```

Positions layout Nodes layouts are algorithms which are used to calculate (x,y) node positions. As they usually are not bind on one specific attribute the *vis:positions* element is placed in the *attributes* element. It documents which algorithm(s) has been used with an optional *referenceURL*. The *vis:layout* element contains a list of parameters which were used. Each *vis:layout* must have a *step* attribute which indicates in which order the layouts were applied.

Listing 59: Positions layout

```
<attributes class="node" mode="static">
    <attribute id="degree" title="Degree" type="integer">
        <default>0</default>
    </attribute>
    <viz:positions>
      <viz:layout algorithm="forceatlas2" referenceURL="https://journals.plos.org/plosone/article?id=10.1371/journal.pone.009
        <viz:param name="scale" type="integer" value="10"/>
        <viz:param name="stronger gravity" type="boolean" value="true"/>
      </viz:layout>
      <viz:layout algorithm="nooverlap" step="2">
        <viz:param name="speed" type="integer" value="3"/>
        <viz:param name="ratio" type="float" value="1.2"/>
        <viz:param name="margin" type="float" value="5.0"/>
      </viz:layout>
    </viz:positions>
</attributes>
```

7 Advices: Parser optimization

This section provides some good tips to write parser-friendly files.