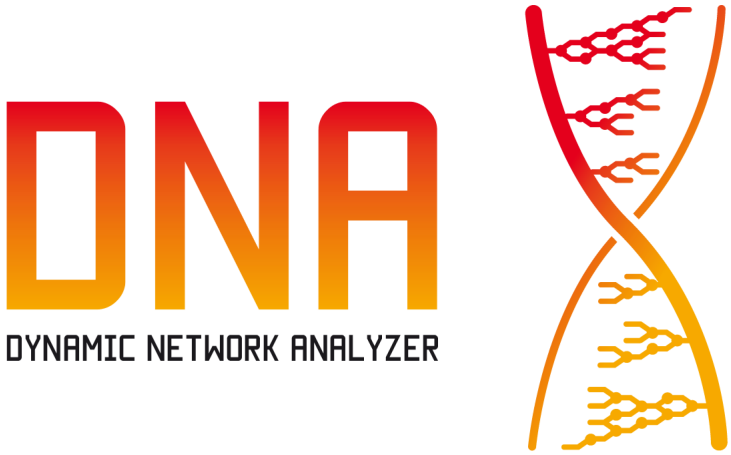

Graph Visualization Instruction Manual

René Wilmes

10.05.2016

Dresden University of Technology

Contents

Chapter 1

Introduction

TODO

Chapter 2

User Interface Structure

TODO

2.1 GraphStyleRules

Chapter 3

Configuration

The graph visualization user interface is designed to be intuitive and easy to use. For most use cases the default configuration will offer a convenient GUI. However, more sophisticated occasions may require to adapt the interface to specific needs. This is achieved by defining custom configuration files via JSON [1]. The general configuration structure is shown in figure 3.1. The following sections will show how to work with the interface and create individual configurations.

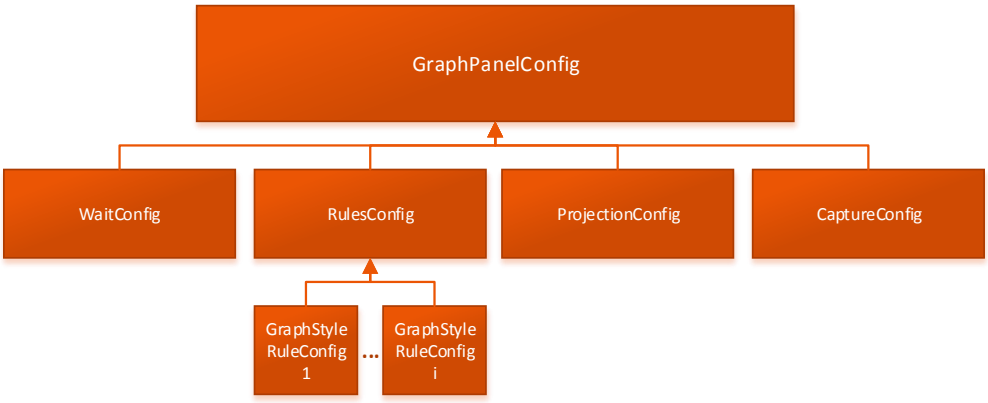

Figure 3.1 – Configuration Structure.

3.1 GraphPanelConfig

The GraphPanelConfig contains the most general configurable aspects as well as additional configuration objects, namely a *WaitConfig*, *ProjectionConfig*, *RulesConfig* and a *CaptureConfig*. The default GraphPanelConfig will be loaded if no other config path is given explicitly prior to initialization. This may be achieved either by setting the respective property via

```
Config.overwrite("GRAPH_VIS_CONFIG_PATH", <config-path>);
```

or by calling the method:

```
GraphVisualization.setConfigPath(<config-path>);
```

All configurable values may be altered during runtime with getter- and setter-methods. In order to change values of subconfigs, e.g. the video record destination directory, one has to retrieve the CaptureConfig from the GraphPanelConfig, which is first retrieved from the GraphPanel directly. This specific case may look like this:

```
GraphPanel panel = GraphVisualization.getCurrentGraphPanel();
GraphPanelConfig config = panel.getGraphPanelConfig();
CaptureConfig ccfg = config.getCaptureConfig();
ccfg.setVideoDir("myvideos/");
```

However, specific values, which are only read and used at initialization, won't have any effect upon change. E.g. the window size of the graph panel is only read when the panel is being initialized. Any modification at a later point won't affect the GUI. All configurable values of the GraphPanelConfig and their descriptions are listed in 3.1.

Value	**Type**	**Description**
Width	Integer	Width of the visualization window.
Height	Integer	Height of the visualization window.
Fullscreen	Boolean	If true the visualization window will be maximized on initialization.
StatPanelEnabled	Boolean	If true the visualization window will contain the StatPanel.
TextPanelEnabled	Boolean	If true the visualization window will contain the TextPanel.
TimetampFormat	String	Sets the used timestamp format.
RenderHQ	Boolean	Enables high-quality rendering.
RenderAA	Boolean	Enables anti-aliasing.
ZoomSpeed	Double	Zoom speed factor.
ScrollSpeed	Double	Scroll speed factor.
ToolTipsEnabled	Boolean	If true tooltips will be enabled.
DirectedEdgeArrowsEnabled	Boolean	If true directed edges will be rendered with arrow-heads.
NodeSize	Double	Default size of nodes.
NodeColor	String	Default color of nodes. All CSS color-keywords are available [2].
EdgeSize	Double	Default size of edges.
Layouter	Enum	Layouter used to layout the graph. Possible values: none, auto, linlog.
AutoLayoutForce	Double	Force of the auto-layouter.
WaitConfig	JSON-Object	A wait config. See 3.1.1.
RulesConfig	JSON-Object	A rules config. See 3.1.2.
ProjectionConfig	JSON-Object	A Projection config. See 3.1.3.
CaptureConfig	JSON-Object	A capture config. See 3.1.4.

Table 3.1 – Configuration values of GraphPanelConfig.

3.1.1 WaitConfig

The WaitConfig object is used to configure special wait times used for smooth graph visualization. When adding multiple nodes and edges at once the layout algorithm may cause a very shaking and unstable view of the graph. Therefore, artificial wait times after each event are used to guarantee a smoother experience. Table 3.2 shows all configurable values.

Value	Type	Description
Enabeld	Boolean	If true wait times are enabled.
EdgeAddition	Long	Wait time after edge additions in ms.
EdgeRemoval	Long	Wait time after edge removals in ms.
EdgeWeightChange	Long	Wait time after edge weight changes in ms.
NodeAddition	Long	Wait time after node additions in ms.
NodeRemoval	Long	Wait time after node removals in ms.
NodeWeightChange	Long	Wait time after node weight changes in ms.

Table 3.2 – Configuration values of WaitConfig.

3.1.2 RulesConfig

The graph visualization allows to define so called *GraphStyleRules* to modify the graph presentation in several ways. See 2.1 for more details. In this section we focus on how to enable and configure said rules using the configuration. A RulesConfig holds multiple configurable *GraphStyleRuleConfis*, each representing a single GraphStyleRule. Each GraphStyleRuleConfig will be identified by a key pointing to the rules class-path. Additionally, there are three shared configurable values for each rule: the *name*, the *enabled*-flag and the *hidden*-flag. Furthermore, one may add additional parameters, which will be used by the respective rules. Note: The way these parameters will be treated is up to the given rule. See the following example 3.2 for an actual RulesConfig definition. The first rule contains all possible configurable values and one additional parameter: *Growth*. The NodeSizeByDegree class increases/decreases each nodes size by the growth-value whenever their degree is being incremented or decremented respectively. The second rule does not contain any values at all. However it will still be enabled.

```
"RulesConfig": {
  "dna.visualization.graph.rules.nodes.NodeSizeByDegree": {
    "Name": "defaultNodeSizeByDegree",
    "Growth": 0.3,
    "Enabled": true,
    "Hidden": false,
  },
  "dna.visualization.graph.rules.nodes.RandomNodeColor": {},
}
```

Figure 3.2 – Example RuleConfig definition.

Value	Type	Description
Name	String	Name of the rule. (If not set *key* will be used as name.)
Enabled	Boolean	If set the rule is enabled. (If not set will be assumed as *true*.)
Hidden	Boolean	If set the rule is hidden from the interface. (If not set will be assumed as *false*.)
Parameter_0	Parameter	First parameter of the respective rule.
Parameter_1	Parameter	Second parameter of the respective rule.
...	...	...
Parameter_i	Parameter	i-th parameter of the respective rule.

Table 3.3 – Configuration values of a GraphStyleRuleConfig.

3.1.3 ProjectionConfig

Since the graphstream library does not support three-dimensional coordinates, two simple projection mechanisms have been implemented to simulate 3D behaviour. In this mode the node weights are assumed as 3D coordinates. For all configurable values see 3.4.

Vanishing Point Projection

This projection mode uses a defined vanishing point and virtually *shifts* all nodes towards the vanishing point. How far the nodes will be shifted depends on the respective z-coordinate. The further a point is away from the screen-plain ($z = 0$) the further it will be shifted. The vanishing point coordinates, a scaling factor and logarithmic scaling can be configured in the ProjectionConfig. Note: If 3d-projection is enabled but vanishing point mode is disabled ortographic projection will be used instead.

Ortographic Projection

The ortographic projection uses a transformation matrix to project the three-dimensional nodes onto the two-dimensional screen-plain, see 3.3. The used matrix and a offset may be defined in the ProjectionConfig. Note: This this mode is only used if the vanishing point projection is disabled.

$$\begin{pmatrix} n_x^{'} \\ n_y^{'} \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \end{pmatrix} \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} + \begin{pmatrix} offset_x \\ offset_y \end{pmatrix}$$

Figure 3.3 – Ortographic projection from node n to two-dimensional node $n^{'}$.

Value	Type	Description
Enabled	Boolean	If true 3d projection will be used.
UseVanishingPoint	Boolean	If true vanishing point will be used
VanishingPointLogScale	Boolean	If true the projection towards the VP will be scaled logarithmic.
VanishingPoint_X	Double	x-coordinate of the VP.
VanishingPoint_Y	Double	y-coordinate of the VP.
VanishingPoint_Z	Double	z-coordinate of the VP.
VanishingPointScaling	Double	Weight of the projection scaling.
ScalingMatrixS0_X	Double	x_0 value of the scaling matrix. For ortographic use only.
ScalingMatrixS0_Y	Double	y_0 value of the scaling matrix. For ortographic use only.
ScalingMatrixS0_Z	Double	z_0 value of the scaling matrix. For ortographic use only.
ScalingMatrixS1_X	Double	x_1 value of the scaling matrix. For ortographic use only.
ScalingMatrixS1_Y	Double	y_1 value of the scaling matrix. For ortographic use only.
ScalingMatrixS1_Z	Double	z_1 value of the scaling matrix. For ortographic use only.
OffsetVector_X	Double	x-value of the offset vector. For ortographic use only.
OffsetVector_Y	Double	y-value of the offset vector. For ortographic use only.

Table 3.4 – Configuration values of a ProjectionConfig.

3.1.4 CaptureConfig

The capture config defines how screenshot and video captures will behave. This includes basic options like destination directory as well as more detailed options, e.g. the frames-per-second included in the video file. For all available configuration values see 3.5.

3.2 Default-Configuration file

The default configuration file located in ”config/gvis/gvis default.cfg” holds default values for each type of object and all possible properties. It represents the ultimate structure a configuration file can assume. When creating manual configuration files missing values will be replaced with the respective values from the default configuration, thus allowing to craft own configurations easily and with little effort. Note that any changes on the default file can prevent the interface from running, for example removing values that are needed as default references.

Value	Type	Description
RecordArea	Enum	Default area to be recorded. Possible values: content, graph, full
VideoDir	String	Destination directory for captured videos.
VideoSuffix	String	Suffix of video files.
VideoFilename	String	Filename of captured video files. If set as ”null” names will be derived from the graph-panel name and (real-time) timestamp.
VideoMaximumLength	Integer	Maximum length of videos in seconds.
VideoRecordFps	Integer	Frames per second recorded from the screen.
VideoFileFps	Integer	Frames per second of the resulting video file.
VideoAutoRecord	Boolean	If set video recording will automatically start upon initialization.
VideoUseCodec	Boolean	If set video will be encoded using the Techsmith codec [3]. Else uncompressed.
VideoBufferImagesOnFs	Boolean	If true images will be buffered on the filesystem.
ScreenshotDir	String	Destination directory for screenshots.
ScreenshotFormat	String	Screenshot format (jpeg, png, bmp, wbmp, gif).
ScreenshotStabilityThreshold	Double	Stability threshold of the layouter when taking screenshots waiting for stability.
ScreenshotStabilityTimeout	Long	Stability timeout for taking screenshots waiting for stability in ms.
ScreenshotForegroundDelay	Long	Artificial delay when taking screenshots to let the machine move the panel to foreground.

Table 3.5 – Configuration values of CaptureConfig.

3.3 Building own configuration files

When creating an own configuration file it is recommended to take the default config structure and fill it with all desired values and settings. The following example 3.4 shows a minimal configuration with the random node sizes and colors. Additionally the size is set to 640x480 and the stat- and text-panels are disabled. See 3.5 for the resulting GraphPanel.

```
{
  "GraphPanelConfig": {
    "Width": 640,
    "Height": 480,

    "StatPanelEnabled": false,
    "TextPanelEnabled": false,

    "RulesConfig": {
      "dna.visualization.graph.rules.nodes.RandomNodeSize": {},
      "dna.visualization.graph.rules.nodes.RandomNodeColor": {},
    }
  }
}
```

Figure 3.4 – Example of a small GraphPanelConfig.

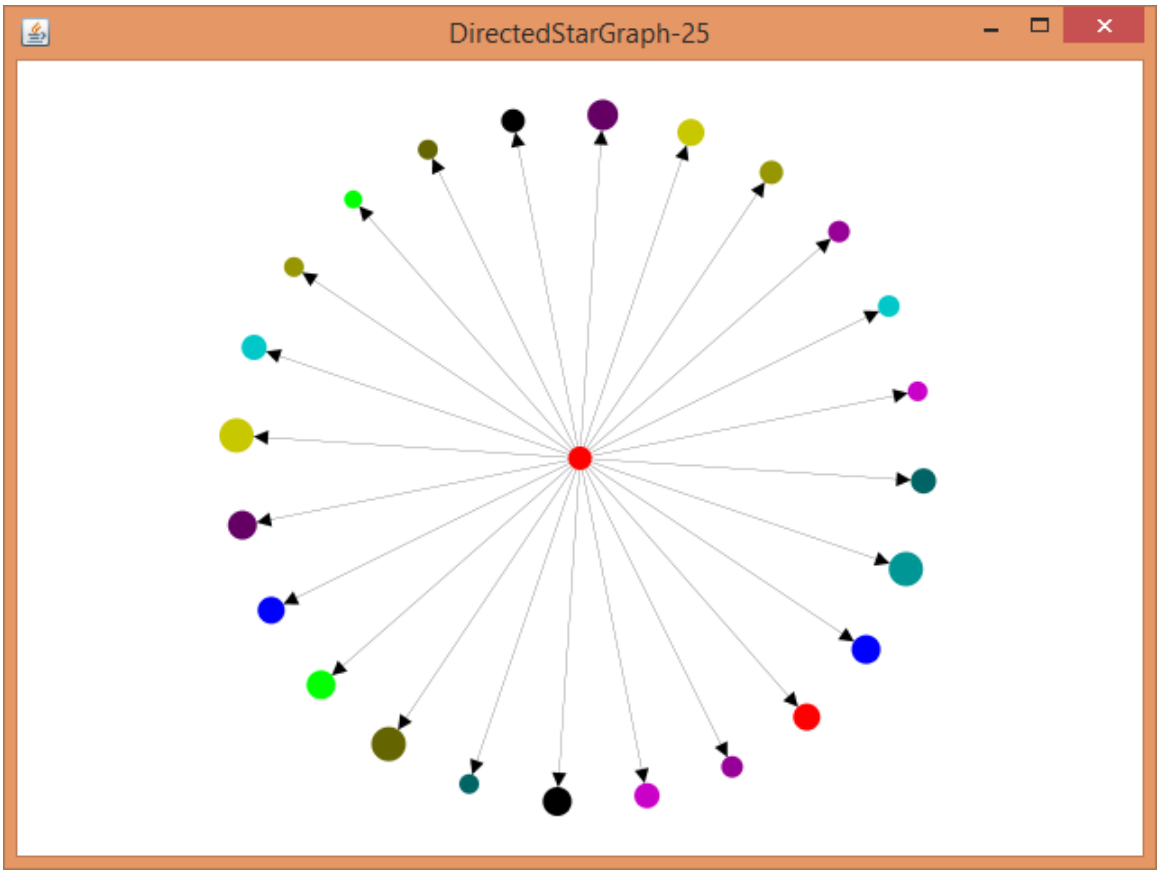

Figure 3.5 – Example visualization of the GraphPanelConfig from 3.4.

Bibliography

[1] Tim Bray, *The javascript object notation (json) data interchange format*, 2014. http://rfc7159.net/rfc7159.

[2] W3schools, *CSS Colors*, http://www.w3schools.com/css/css_colors.asp.

[3] TechSmith, *TSCC-codec*, https://www.techsmith.com/codecs.html.