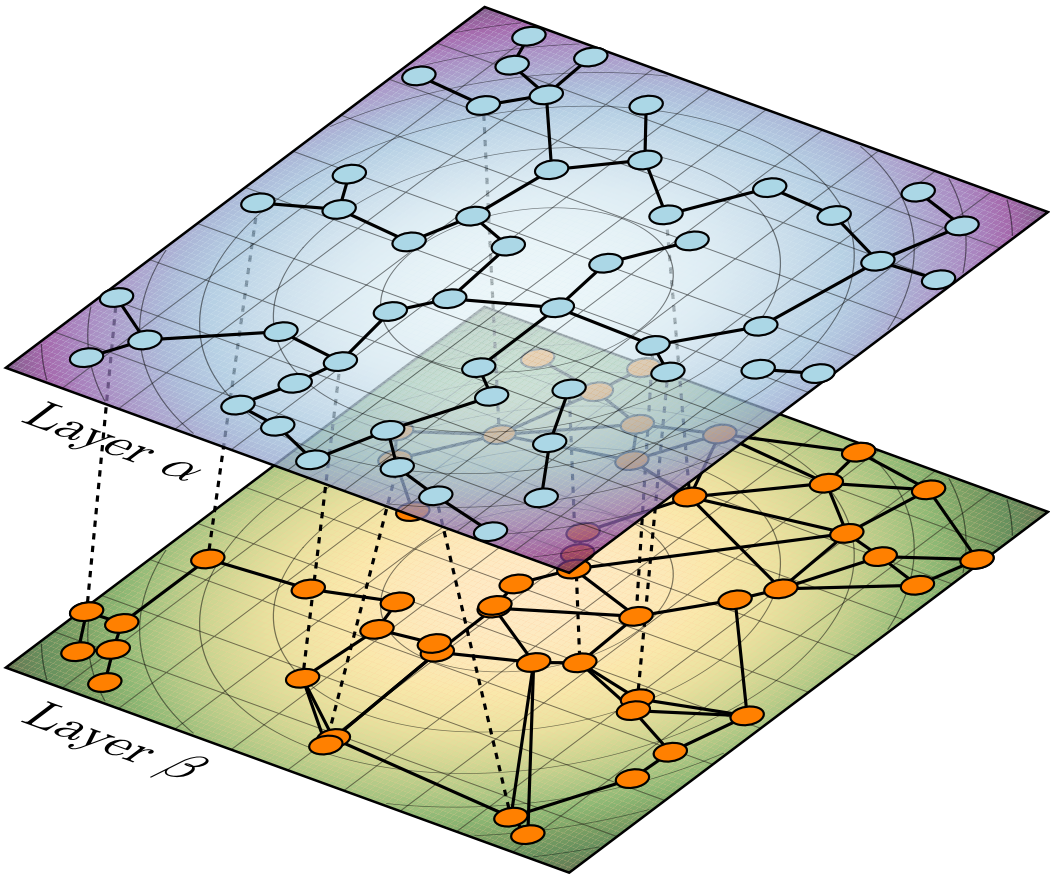
JÜRGEN HACKL

TKZ-NETWORK MANUAL



VERSION 0.1

The tkz-network package is still under development. Hence, changes in the commands and functionality cannot be excluded.

VERSION 0.1

https://github.com/hackl/tkz-network

Released, September 2017

Contents

1 Introduction

In recent years, complex network theory becomes more and more popular within the scientific community. Besides a solid mathematical base on which these theories are built on, a visual representation of the networks allow communicating complex relationships to a broad audience.

Nowadays, a variety of great visualization tools are available, which helps to structure, filter, manipulate and of course to visualize the networks. However, they come with some limitations, including the need for specific software tools, difficulties to embed the outputs properly in a LaTeX file (e.g. font type, font size, additional equations and math symbols needed, ...) and challenges in the post-processing of the graphs, without rerunning the software tools again.

In order to overcome this issues, the package `tkz-network` was created. Some of the features are:

- LaTeX is a standard for scientific publications and widely used
- beside LaTeX no other software is needed
- no programming skills are needed
- simple to use but allows 100% control over the output
- easy for post-processing (e.g. adding drawings, texts, equations,...)
- same fonts, font sizes, mathematical symbols, ... as in the document
- no quality loss of the output due to the pdf format
- networks are easy to adapt or modify for lectures or small examples
- able to visualize larger networks
- three-dimensional visualization of (multilayer) networks
- compatible with other visualization tools

1.1 How to read this manual?

The aim of this manual is to describe the use of the `tkz-network` library for visualizing networks. To ensure an easy use of the elements and to keep the clarity, this manual is structured as follows:

- In Chapter 2 the elements to create simple networks (by hand) in a plane are explained. Thereby, the use of the commands `\Vertex` and `\Edge` are shown.
- How to create complex networks from external files[1] are explained in Chapter 3. The main commands, therefore are `\Vertices` and `\Edges` which are using the same options as in the simple case.
- In Chapter 4, the visualization of multilayer networks is explained. Additional visualization tools such as `\Plain` and `Layer` are introduced.
- The default settings used and how they can be modified is explained in Chapter 5.
- Information about troubleshooting and support is given in Chapter 6
- Since this is the alpha version (0.1) of the package, features which will be probably added and commands which have to be fixed are listed in Appendix A.

[1] e.g. `igraph` or `networkx`

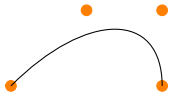
1.1.1 A few explanations

The images in this manual are created with the `tkz-network` library or Ti*k*Z . The code used for this is specified for each image.

```
\begin{tikzpicture}
  \filldraw (-.2,.2) circle (2pt) (.2,.2) circle (2pt);
  \draw (0,0) circle (5mm) (-.3,-.1) .. controls (0,-.3) ..
      (.3,-.1);
\end{tikzpicture}
```

Special additions which are needed for a better understanding are shown in orange but are not in the sample code available.

```
\begin{tikzpicture}
  \draw (0,0) .. controls (1,1) and (2,1) .. (2,0);
\end{tikzpicture}
```



1.1.2 Inputs

The commands in the `tkz-network` library (e.g. `\Vertex`, `\Edge`) always start with capital letters and DO NOT need a semicolon «;» at the end. Boolean arguments start also with capital letters (e.g. ⟨*NoLabel*⟩). Arguments which need an user input, use are written in small letters (e.g. ⟨*color*⟩).

Basically, one can distinguish between the mandatory argument { } and the optional argument [ ]. The first values must be entered compulsory. By contrast, nothing has to be entered for the optional input. Additional features (e.g. ⟨*size*⟩)) can be activated when entering optional parameters.

When entering size values the base unit is always predefined in [*cm*][2], except for line widths which are dedined in [*pt*]. Percentage values % are always specified as decimal values; for example, 100% = 1.0 and 10% corresponds to 0.1.

[2] The default unit can be changed with `\SetDefaultUnit`; see Section 5.1

1.1.3 *Additional help*

Is the manual not enough, occur some ambiguities or some Ti*k*Z commands are unclear, please have a look in the "Ti*k*Z and PGF Manual" from Till Tantau[3].

Should you have any further questions, please do not hesitate to contact me.

[3] `http://mirror.switch.ch/ftp/mirror/tex/graphics/pgf/base/doc/pgfmanual.pdf`

1.2 *Installation*

Actually, we can hardly speak of an installation since only the necessary package `tkz-network` must be loaded in the preamble of your document.

The current release of the package is available via CTAN[4]. A release candidate for the next version of `tkz-network` is available on github[5]

Is the package installed or the style file is stored in the folder of the main file, so the library can be imported, as the following example shows:

[4] **TODO!** upload the package to CTAN, and add here the link

[5] `https://github.com/hackl/`

```
% ------------
% header
\documentclass{scrreprt}

% ------------
% packages
\usepackage{tkz-network}
```

1.3 *Additional necessary packages*

To use all commands and options of Ti*k*Z , possibly some packages need to be reloaded. These missing files (or their names) appear in the error log when you convert the file. However, for the package described in this manual, it is sufficient to use the library and the Ti*k*Z standard commands.

2 Simple Networks

2.1 Vertex

On essential command is `\Vertex`, which allow placing vertices in the document and modify their appearance.

`\Vertex[`⟨*local options*⟩`]{`*Name*`}`

In order to be able to place a vertex, a non-empty *Name* argument is required. This argument defines the vertex's reference name, which must be unique. Mathematical symbols are not allowed as name as well as no blank spaces. The *Name* should not be confused with the ⟨*label*⟩, that is used for display; for example one may want to display A_1 while the name will be *A1*.

For a `\Vertex` the following options are available:

Option	Default	Type	Definition
x	0	measure	x-coordinate
y	0	measure	y-coordinate
size	{}	measure	diameter of the circle
color	{}	color	fillcolor of vertex
opacity	{}	number	opacity of the fill color
label	{}	string	label
position	center	value[a]	label position
distance	0	measure	label distance from the center
style	{}	string	additional TikZ styles
layer	{}	number	assigned layer of the vertex
NoLabel	false	boolean	delete the label
IdAsLabel	false	boolean	uses the *Name* as label
Math	false	boolean	displays the label in math mode
RGB	false	boolean	allow RGB colors

[a] either measure or string

Table 2.1: Local options for the `\Vertex` command.

The order how the options are entered does not matter. Changes to the default Vertex layout can be made with `\SetVertexStyle`[1]

[1] see Section 5.2

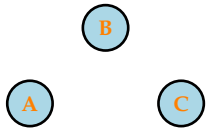
`\Vertex[`⟨*x*⟩`=`*measure*`,`⟨*y*⟩`=`*measure*`]{`*Name*`}`

The location of the vertices are determined by Cartesian coordinates in ⟨*x*⟩ and ⟨*y*⟩. The coordinates are optional. If no coordinates are determined the vertex will be placed at the origin $(0, 0)$. The entered *measure* are in default units (cm). Changing the unites (locally) can be done by adding the unit to the *measure*[2]. Changes to the default setting can be made with `\SetDefaultUnit`[3].

[2] e.g. x=1 in

[3] see Section 5.1

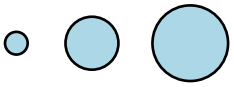
```
\begin{tikzpicture}
  \Vertex{A}
  \Vertex[x=1,y=1]{B}
  \Vertex[x=2]{C}
\end{tikzpicture}
```



`\Vertex[`⟨*size*⟩`=`*measure*`]{`*Name*`}`

The diameter of the vertex can be changed with the option ⟨*size*⟩. Per default a vertex has 0.6 cm in diameter. Also, here the default units are cm and have not to be added to the *measure*.

```
\begin{tikzpicture}
  \Vertex[size=.3]{A}
  \Vertex[x=1,size=.7]{B}
  \Vertex[x=2.3,size=1]{C}
\end{tikzpicture}
```



`\Vertex[`⟨*color*⟩`=`*color*`]{`*Name*`}`

To change the fill color of each vertex individually, the option ⟨*color*⟩ has to be used. Without the option ⟨*RGB*⟩ set, the default Ti*k*Z and LaTeX colors can be applied.

```
\begin{tikzpicture}
  \Vertex[color = blue]{A}
  \Vertex[x=1,color=red]{B}
  \Vertex[x=2,color=green!70!blue]{C}
\end{tikzpicture}
```

`\Vertex[`⟨*opacity*⟩`=`*number*`]{`*Name*`}`

With the option ⟨*opacity*⟩ the opacity of the vertex fill color can be modified. The range of the *number* lies between 0 and 1. Where 0 represents a fully transparent fill and 1 a solid fill.

```
\begin{tikzpicture}
  \Vertex[opacity = 1]{A}
  \Vertex[x=1,opacity =.7]{B}
  \Vertex[x=2,opacity =.2]{C}
\end{tikzpicture}
```

`\Vertex[`⟨*label*⟩`=`*string*`]{`*Name*`}`

In `tkz-network` there are several ways to define the labels of the vertices and edges. The common way is via the option ⟨*label*⟩. Here, any *string* argument can be used, including blank spaces. The environment $ $ can be used to display mathematical expressions.

```
\begin{tikzpicture}
  \Vertex[label=foo]{A}
  \Vertex[x=1,label=bar]{B}
  \Vertex[x=2,label=$u_1$]{B}
\end{tikzpicture}
```

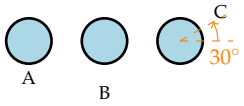
`\Vertex[`⟨*label*⟩`=`*string*`,`⟨*position*⟩`=`*value*`,`⟨*distance*⟩`=`*number*`]{`*Name*`}`

Per default the ⟨*position*⟩ of the ⟨*label*⟩ is in the *center* of the vertex. Classical TikZ commands[4] can be used to change the ⟨*position*⟩ of the ⟨*label*⟩. Instead, using such command, the position can be determined via an angle, by entering a *number* between −360 and 360. The origin (0°) is the *y* axis. A positive *number* change the ⟨*position*⟩ counter clockwise, while a negative *number* make changes clockwise.

[4] e.g. *above*, *below*, *left*, *right*, *above left*, *above right*,...

With the option, ⟨*distance*⟩ the distance between the vertex and the label can be changed.

```
\begin{tikzpicture}
  \Vertex[label=A,position=below]{A}
  \Vertex[x=1,label=B,position=below,distance=2mm]{B}
  \Vertex[x=2,label=C,position=30,distance=1mm]{C}
\end{tikzpicture}
```



`\Vertex[`⟨*style*⟩`={`*string*`}]{`*Name*`}`

Any other TikZ style option or command can be entered via the option ⟨*style*⟩. Most of these commands can be found in the "TikZ and PGF Manual". Contain the commands additional options (e.g.⟨*shading*⟩=*ball*), then the argument for the ⟨*style*⟩ has to be between { } brackets.

```
\begin{tikzpicture}
  \Vertex[style={color=green}]{A}
  \Vertex[x=1,style=dashed]{B}
  \Vertex[x=2,style={shading=ball}]{C}
\end{tikzpicture}
```

`\Vertex[`⟨*IdAsLabel*⟩`]{`*Name*`}`

`\Vertex[`⟨*NoLabel*⟩`,`⟨*label*⟩`=`*string*`]{`*Name*`}`

⟨*IdAsLabel*⟩ is a boolean option which assigns the *Name* of the vertex as label. On the contrary, ⟨*NoLabel*⟩ suppress all labels.

```
\begin{tikzpicture}
  \Vertex[IdAsLabel]{A}
  \Vertex[x=1,label=B,NoLabel]{B}
  \Vertex[x=2,IdAsLabel,NoLabel]{C}
\end{tikzpicture}
```

`\Vertex[`⟨*Math*⟩`,`⟨*label*⟩`=`*string*`]{`*Name*`}`

The option ⟨*Math*⟩ allows transforming labels into mathematical expressions without using the $ $ environment. In combination with ⟨*IdAsLabel*⟩ allows this option also mathematical expressions by the definition of the vertex *Name*.

```
\begin{tikzpicture}
  \Vertex[IdAsLabel]{A1}
  \Vertex[x=1,label=B_1,Math]{B}
  \Vertex[x=2,Math,IdAsLabel]{C_1}
\end{tikzpicture}
```

`\Vertex[`⟨*RGB*⟩`,`⟨*color*⟩`=`*RGB values*`]{`*Name*`}`

In order to display RGB colors for the vertex fill color, the option ⟨*RGB*⟩ has to be entered. In combination with this option, the ⟨*color*⟩ hast to be a list with the *RGB values*, separated by «,» and within { }.[5]

[5] e.g. the RGB code for white: {255, 255, 255}

```
\begin{tikzpicture}
  \Vertex[RGB,color={127,201,127}]{A}
  \Vertex[x=1,RGB,color={190,174,212}]{B}
  \Vertex[x=2,RGB,color={253,192,134}]{C}
\end{tikzpicture}
```

`\Vertex[`⟨*layer*⟩`=`*number*`]{`*Name*`}`

With the option ⟨*layer*⟩ the vertex can be assigned to a specific layer. More about this option and the use of layers is explained in Chapter 4.

2.2 Edge

The second essential command is an `\Edge`, which allow connecting two vertices.

`\Edge[`⟨*local options*⟩`](`*Vertex i*`)(`*Vertex j*`)`

Edges can be generated between one or two vertices. In the first case, a self-loop will be generated. As mandatory arguments the *Names* of the vertices which should be connected must be entered between ( ) brackets. In case of a directed edge, the order is important. An edge is created from *Vertex i* (origin) to *Vertex j* (destination).

For an `\Edge` the following options are available:

Option	Default	Type	Definition
lw	{}	measure	line width of the edge
color	{}	color	edge color
opacity	{}	number	opacity of the edge
bend	0	number	angle out/in of the vertex
label	{}	string	label
position	{}	string	label position
distance	0.5	number	label distance from Vertex i
style	{}	string	additional TikZ styles
loopsize	1cm	measure	size parameter of the self-loop
loopposition	0	number	orientation of the self-loop
loopshape	90	number	loop angle out/in of the vertex
Direct	false	boolean	allow directed edges
Math	false	boolean	displays the label in math mode
RGB	false	boolean	allow RGB colors

Table 2.2: Local options for the `\Edge` command.

The options ⟨*loopsize*⟩, ⟨*loopposition*⟩, and ⟨*loopsize*⟩ are only for self-loops available.

`\Edge(`*Vertex i*`)(`*Vertex j*`)`

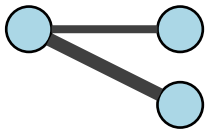
An edge is created between *Vertex i* and *Vertex j*.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge(A)(B)
\end{tikzpicture}
```

`\Edge[`⟨*lw*⟩`=`*measure*`](`*Vertex i*`)(`*Vertex j*`)`

The line width of an edge can be modified with the option ⟨*lw*⟩. Here, the unit of the *measure* has to be specified. The default value is 1.5 pt.

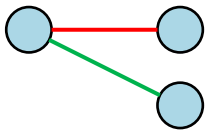
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B} \Vertex[x=2,y=-1]{C}
  \Edge[lw=3pt](A)(B)
  \Edge[lw=5pt](A)(C)
\end{tikzpicture}
```



`\Edge[`⟨*color*⟩`=`*color*`](`*Vertex i*`)(`*Vertex j*`)`

To change the line color of each edge individually, the option ⟨*color*⟩ has to be used. Without the option ⟨*RGB*⟩ set, the default TikZ and LaTeX colors can be applied.

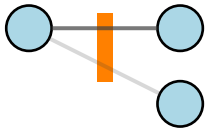
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B} \Vertex[x=2,y=-1]{C}
  \Edge[color=red](A)(B)
  \Edge[color=green!70!blue](A)(C)
\end{tikzpicture}
```



`\Edge[`⟨*opacity*⟩`=`*number*`](`*Vertex i*`)(`*Vertex j*`)`

With the option ⟨*opacity*⟩ the opacity of the edge line can be modified. The range of the *number* lies between 0 and 1. Where 0 represents a fully transparent fill and 1 a solid fill.

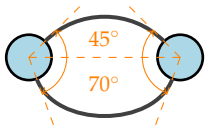
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B} \Vertex[x=2,y=-1]{C}
  \Edge[opacity=.7](A)(B)
  \Edge[opacity=.2](A)(C)
\end{tikzpicture}
```



`\Edge[`⟨*bend*⟩`=`*number*`](`*Vertex i*`)(`*Vertex j*`)`

The shape of the edge can be modified with the ⟨*bend*⟩ option. If nothing is specified a straight edge, between the vertices, is drawn. The *number* defines the angle in which the edge is diverging from its straight connection. A positive *number* bend the edge counter clockwise, while a negative *number* make changes clockwise.

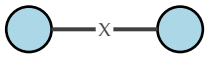
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge[bend=45](A)(B)
  \Edge[bend=-70](A)(B)
\end{tikzpicture}
```



`\Edge[`⟨*label*⟩`=`*string*`](`*Vertex i*`)(`*Vertex j*`)`

An edge is labeled with the option ⟨*label*⟩. For the label any *string* argument can be used, including blank spaces. The environment $ $ can be used to display mathematical expressions.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge[label=X](A)(B)
\end{tikzpicture}
```

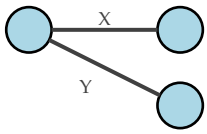


`\Edge[`⟨*label*⟩`=`*string*`,`⟨*position*⟩`=`*string*`](`*Vertex i*`)(`*Vertex j*`)`

Per default the ⟨*label*⟩ is positioned in between both vertices in the center of the line. Classical TikZ commands[6] can be used to change the ⟨*position*⟩ of the ⟨*label*⟩.

[6] e.g. *above*, *below*, *left*, *right*, *above left*, *above right*,...

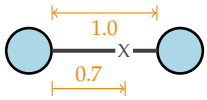
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B} \Vertex[x=2,y=-1]{C}
  \Edge[label=X,position=above](A)(B)
  \Edge[label=Y,position={below left=2mm}](A)(C)
\end{tikzpicture}
```



`\Edge[`⟨*label*⟩`=`*string*`,`⟨*distance*⟩`=`*number*`](`*Vertex i*`)(`*Vertex j*`)`

The label position between the vertices can be modified with the ⟨*distance*⟩ option. Per default the ⟨*label*⟩ is centered between both vertices. The position is expressed as the percentage of the length between the vertices, e.g. of ⟨*distance*⟩=0.7, the label is placed at 70% of the edge length away of *Vertex i*.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge[label=X,distance=.7](A)(B)
\end{tikzpicture}
```



`\Edge[`⟨*style*⟩`=`*string*`](`*Vertex i*`)(`*Vertex j*`)`

Any other TikZ style option or command can be entered via the option ⟨*style*⟩. Most of these commands can be found in the "TikZ and PGF Manual". Contain the commands additional options (e.g.⟨*shading*⟩=*ball*), then the argument for the ⟨*style*⟩ has to be between { } brackets.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge[style={dashed}](A)(B)
\end{tikzpicture}
```

`\Edge(`*Vertex i*`)(`*Vertex i*`)`

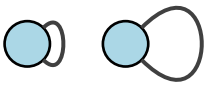
Self-loops are created by using the same vertex as origin and destination. Beside the options explained above, there are three self-loop specific options: ⟨*loopsize*⟩, ⟨*loopposition*⟩, and ⟨*loopshape*⟩.

```
\begin{tikzpicture}
  \Vertex{A}
  \Edge(A)(A)
\end{tikzpicture}
```

`\Edge[`⟨*loopsize*⟩`=`*measure*`](`*Vertex i*`)(`*Vertex i*`)`

With the option ⟨*loopsize*⟩ the length of the edge can be modified. The *measure* value has to be insert together with its units. Per default the ⟨*loopsize*⟩ is 1 cm.

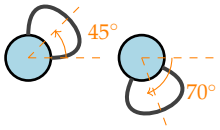
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=1.3]{B}
  \Edge[loopsize=.5cm](A)(A)
  \Edge[loopsize=1.5cm](B)(B)
\end{tikzpicture}
```



`\Edge[`⟨*loopposition*⟩`=`*number*`](`*Vertex i*`)(`*Vertex i*`)`

The position of the self-loop is defined via the rotation angle around the vertex. The origin (0°) is the *y* axis. A positive *number* change the ⟨*loopposition*⟩ counter clockwise, while a negative *number* make changes clockwise.

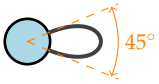
```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=1.5]{B}
  \Edge[loopposition=45](A)(A)
  \Edge[loopposition=-70](B)(B)
\end{tikzpicture}
```



`\Edge[`⟨*loopshape*⟩`=`*number*`](`*Vertex i*`)(`*Vertex i*`)`

The shape of the self-loop is defined by the enclosing angle. The shape can be changed by decreasing or increasing the argument value of the ⟨*loopshape*⟩ option.

```
\begin{tikzpicture}
  \Vertex{A}
  \Edge[angle=45](A)(A)
\end{tikzpicture}
```



`\Edge[`⟨*Direct*⟩`](`*Vertex i*`)(`*Vertex j*`)`

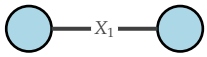
Directed edges are created by enabling the option ⟨*Direct*⟩. The arrow is drawn from *Vertex i* to *Vertex j*.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge[Direct](A)(B)
\end{tikzpicture}
```

`\Edge[Math, label=`⟨*string*⟩`](`*Vertex i*`)(`*Vertex j*`)`

The option ⟨*Math*⟩ allows transforming labels into mathematical expressions without using the $ $ environment.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B}
  \Edge[Math,label=X_1](A)(B)
\end{tikzpicture}
```

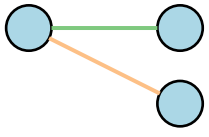


`\Edge[RBG, color=`⟨*RGB value*⟩`](`*Vertex i*`)(`*Vertex j*`)`

In order to display RGB colors for the line color of the edge, the option ⟨*RGB*⟩ has to be entered. In combination with this option, the ⟨*color*⟩ hast to be a list with the *RGB values*, separated by «,» and within { }.[7]

[7] e.g. the RGB code for white: {255, 255, 255}

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=2]{B} \Vertex[x=2,y=-1]{C}
  \Edge[RGB,color={127,201,127}](A)(B)
  \Edge[RGB,color={253,192,134}](A)(C)
\end{tikzpicture}
```



3 Complex Networks

While in Chapter 2 the building blocks of the networks are introduced, here the main strength of the `tkz-network` package is explained. This includes creating networks based on data, obtained from other sources (e.g. Python, R, GIS). The idea is that the layout will be done by this external sources and `tkz-network` is used make some changes and to recreate the networks in LaTeX.

3.1 Vertices

The `\Vertices` command is the extension of the `\Vertex` command. Instead of a single vertex, a set of vertices will be drawn. This set of vertices is defined in an external file but can be modified with `\Vertices`.

`\Vertices[`⟨*global options*⟩`]{`*filename*`}`

The vertices have to be stored in a clear text file[1], preferentially in a `.csv` format. The first row should contain the headings, which are equal to the options defined in Table 2.1. Option are separated by a comma «,». Each new row is corresponds to a new vertex.

[1] e.g. .txt, .tex, .csv, .dat, ...

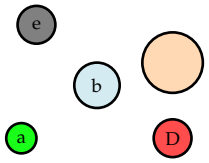
File: vertices.csv

```
id, x, y ,size,color ,opacity,label,IdAsLabel,NoLabel
 A, 0,  0, .4 ,green ,  .9   ,  a  ,  false  , false
 B, 1, .7, .6 ,      ,  .5   ,  b  ,  false  , false
 C, 2,  1, .8 ,orange,  .3   ,  c  ,  false  , true
 D, 2,  0, .5 ,red   ,  .7   ,  d  ,  true   , false
 E,.2,1.5, .5 ,gray  ,       ,  e  ,  false  , false
```

Only the ⟨*id*⟩ value is mandatory for a vertex and corresponds to the *Name* argument of a single `\Vertex`. Therefore, the same rules and naming conventions apply as for the *Name* argument: no mathematical expressions, no blank spaces, and the ⟨*id*⟩ must be unique! All other options are optional. No specific order of the options must be maintained. If no value is entered for an option, the default value will be chosen[2]. The *filename* should not contain blank spaces or special characters. The vertices are drawn by the command `\Vertex` with the *filename* plus file format (e.g. `.csv`). If the vertices file is not in the same directory as the main LaTeX file, also the path has to be specified.

[2] **TODO!** This is NOT valid for Boolean options, here values for all vertices have to be entered.

```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
\end{tikzpicture}
```



Predefined `\Vertex` options can be overruled by the ⟨*global options*⟩ of the `\Vertices` command; I.e. these options apply for all vertices in the file. For the `\Vertices` the following options are available:

Option	Default	Type	Definition
size	{}	measure	diameter of the circles
color	{}	color	fillcolor of vertices
opacity	{}	number	opacity of the fill color
style	{}	string	additional TikZ styles
layer	{}	number	assigned layer of the vertices
NoLabel	false	boolean	delete the labels
IdAsLabel	false	boolean	uses the *Names* as labels
Math	false	boolean	displays the labels in math mode
RGB	false	boolean	allow RGB colors

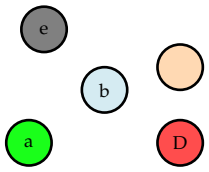
Table 3.1: Global options for the `\Vertices` command.

The use of these options are similar to the options for a single `\Vertex` defined in Section 2.1.

`\Vertices[`⟨*size*⟩`=`*measure*`]{`*filename*`}`

The diameter of the vertices can be changed with the option ⟨*size*⟩. Per default a vertex has 0.6 cm in diameter. Also, here the default units are cm and have not to be added to the *measure*.

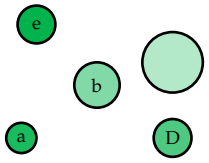
```
\begin{tikzpicture}
  \Vertices[size=.6]{data/vertices.csv}
\end{tikzpicture}
```



`\Vertices[`⟨*color*⟩`=`*color*`]{`*filename*`}`

To change the fill color for all vertices, the option ⟨*color*⟩ has to be used. Without the option ⟨*RGB*⟩ set, the default TikZ and LaTeX colors can be applied.

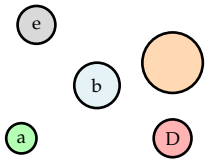
```
\begin{tikzpicture}
  \Vertices[color=green!70!blue]{data/vertices.csv}
\end{tikzpicture}
```



`\Vertices[`⟨*opacity*⟩`=`*number*`]{`*filename*`}`

With the option ⟨*opacity*⟩ the opacity of all vertices fills colors can be modified. The range of the *number* lies between 0 and 1. Where 0 represents a fully transparent fill and 1 a solid fill.

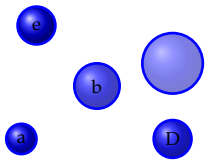
```
\begin{tikzpicture}
  \Vertices[opacity=.3]{data/vertices.csv}
\end{tikzpicture}
```



`\Vertices[`⟨*style*⟩`=`*string*`]{`*filename*`}`

Any other TikZ style option or command can be entered via the option ⟨*style*⟩. Most of these commands can be found in the "TikZ and PGF Manual". Contain the commands additional options (e.g.⟨*shading*⟩=*ball*), then the argument for the ⟨*style*⟩ has to be between { } brackets.

```
\begin{tikzpicture}
  \Vertices[style={shading=ball,blue}]{data/vertices.csv}
\end{tikzpicture}
```

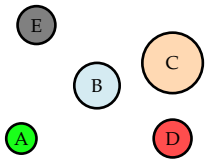


`\Vertices[`⟨*IdAsLabel*⟩`]{`*filename*`}`

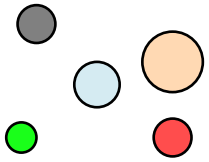
`\Vertices[`⟨*NoLabel*⟩`]{`*filename*`}`

⟨*IdAsLabel*⟩ is a boolean option which assigns the ⟨*id*⟩ of the single vertices as labels. On the contrary, ⟨*NoLabel*⟩ suppress all labels.

```
\begin{tikzpicture}
  \Vertices[IdAsLabel]{data/vertices.csv}
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \Vertices[NoLabel]{data/vertices.csv}
\end{tikzpicture}
```



`\Vertices[`⟨*RGB*⟩`]{`*filename*`}`

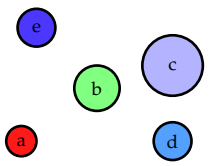
In order to display RGB colors for the vertex fill colors, the option ⟨*RGB*⟩ has to be entered. Additionally, the RGB values have to be specified in the file where the vertices are stored. Each value has its own column with the caption ⟨*R*⟩, ⟨*G*⟩, and ⟨*B*⟩.

```
id, x, y ,size, color,opacity,label, R , G , B
 A, 0,  0, .4 , green,  .9   ,  a  ,255,  0,  0
 B, 1, .7, .6 ,      ,  .5   ,  b  ,  0,255,  0
 C, 2,  1, .8 ,orange,  .3   ,  c  ,  0,  0,255
 D, 2,  0, .5 ,   red,  .7   ,  d  , 10,120,255
 E,.2,1.5, .5 ,  gray,       ,  e  , 76, 55,255
```

File: vertices_RGB.csv

The "normal" color definition can also be part of the vertex definition. If the option ⟨*RGB*⟩ is not set, then the colors under ⟨*color*⟩ are applied.

```
\begin{tikzpicture}
  \Vertices[RGB]{data/vertices_RGB.csv}
\end{tikzpicture}
```



`\Vertices[`⟨*layer*⟩`=`*number*`]{`*filename*`}`

With the option ⟨*layer*⟩, only the vertices on the selected layer are plotted. More about this option and the use of layers is explained in Chapter 4.

3.2 *Edges*

The `\Edges` command is the extension of the `\Edge` command. Instead of a single edge, a set of edges will be drawn. This set of edges is defined in an external file but can be modified with `\Edges`.

`\Edges[`⟨*global options*⟩`]{`*filename*`}`

Like the vertices, the edges have to be stored in a clear text file[3], preferentially in a `.csv` format. The first row should contain the headings, which are equal to the options defined in Table 2.2. Option are separated by a comma «,». Each new row is corresponds to a new edge.

[3] e.g. .txt, .tex, .csv, .dat, ...

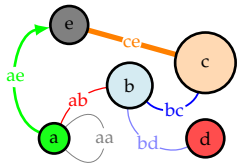
File: edges.csv

```
u,v,label,lw,color ,opacity,bend, R , G , B ,Direct
A,B, ab  ,.5,red   ,   1   ,  30,  0,120,255,false
B,C, bc  ,.7,blue  ,   1   , -60, 76, 55,255,false
B,D, bd  ,.5,blue  ,  .5   , -60, 76, 55,255,false
A,E, ae  , 1,green ,   1   ,  75,255,  0,  0,true
C,E, ce  , 2,orange,   1   ,   0,150,150,150,false
A,A, aa  ,.3,black ,  .5   ,  75,255,  0  ,0,false
```

The mandatory values are the ⟨*u*⟩ and ⟨*v*⟩ argument, which corresponds to the *Vertex i* and *Vertex j* arguments of a single `\Edge`. Edges can only create if a vertex exists with the same *Name*. All other options are optional. No specific order of the options must be maintained. If no value is entered for an option, the default value will be chosen[4]. The *filename* should not contain blank spaces or special characters. The edges are drawn by the command `\Edges` with the *filename* plus file format (e.g. `.csv`). If the edges file is not in the same directory as the main LaTeX file, also the path has to be specified. In order to draw edges, first, the vertices have to be generated. Only then, edges can be assigned.

[4] **TODO!** This is NOT valid for Boolean options, here values for all vertices have to be entered.

```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges{data/edges.csv}
\end{tikzpicture}
```



Predefined `\Edge` options can be overruled by the ⟨*global options*⟩ of the `\Edges` command; I.e. these options apply for all edges in the file. For the `\Edges` the following options are available:

Option	Default	Type	Definition
lw	{}	measure	line width of the edge
color	{}	color	edge color
opacity	{}	number	opacity of the edge
style	{}	string	additional TikZ styles
vertices	{}	file	vertices were the edges are assigned to
layer	{}	number	edges in specific layers
Direct	false	boolean	allow directed edges
Math	false	boolean	displays the labels in math mode
NoLabel	false	boolean	delete the labels
RGB	false	boolean	allow RGB colors
NotInBG	false	boolean	edges are not in the background layer

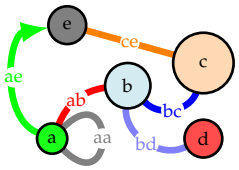
Table 3.2: Global options for the `\Edges` command.

The use of these options are similar to the options for a single `\Edge` defined in Section 2.2.

`\Edges[`⟨*lw*⟩`=`*measure*`]{`*filename*`}`

The line width of the edges can be modified with the option ⟨*lw*⟩. Here, the unit of the *measure* can be specified, otherwise, it is in pt.

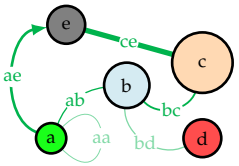
```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[lw=2.5]{data/edges.csv}
\end{tikzpicture}
```



`\Edges[`⟨*color*⟩`=`*color*`]{`*filename*`}`

To change the line color of all edges, the option ⟨*color*⟩ has to be used. Without the option ⟨*RGB*⟩ set, the default TikZ and LaTeX colors can be applied.

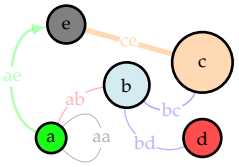
```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[color=green!70!blue]{data/edges.csv}
\end{tikzpicture}
```



`\Edges[`⟨*opacity*⟩`=`*number*`]{`*filename*`}`

With the option ⟨*opacity*⟩ the opacity of all edge lines can be modified. The range of the *number* lies between 0 and 1. Where 0 represents a fully transparent fill and 1 a solid fill.

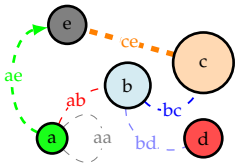
```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[opacity=0.3]{data/edges.csv}
\end{tikzpicture}
```



`\Edges[`⟨*style*⟩`=`*string*`]{`*filename*`}`

Any other TikZ style option or command can be entered via the option ⟨*style*⟩. Most of these commands can be found in the "TikZ and PGF Manual".

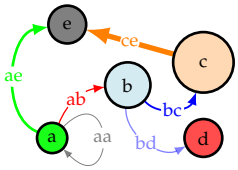
```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[style={dashed}]{data/edges.csv}
\end{tikzpicture}
```



`\Edges[`⟨*Direct*⟩`]{`*filename*`}`

Directed edges are created by enabling the option ⟨*Direct*⟩. The arrow is drawn from ⟨*u*⟩ to ⟨*v*⟩.

```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[Direct]{data/edges.csv}
\end{tikzpicture}
```



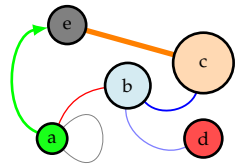
`\Edges[Math]{`*filename*`}`

The option ⟨*Math*⟩ allows transforming labels into mathematical expressions without using the $ $ environment.

`\Edges[`⟨*NoLabel*⟩`]{`*filename*`}`

The option ⟨*NoLabel*⟩ suppress all edge labels.

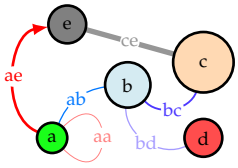
```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[NoLabel]{data/edges.csv}
\end{tikzpicture}
```



`\Edges[`⟨*RGB*⟩`]{`*filename*`}`

In order to display RGB colors for the edge line colors, the option ⟨*RGB*⟩ has to be entered. Additionally, the RGB values have to be specified in the file where the vertices are stored. Each value has its own column with the caption ⟨*R*⟩, ⟨*G*⟩, and ⟨*B*⟩. The "normal" color definition can also be part of the vertex definition. If the option ⟨*RGB*⟩ is not set, then the colors under ⟨*color*⟩ are applied.

```
\begin{tikzpicture}
  \Vertices{data/vertices.csv}
  \Edges[RGB]{data/edges.csv}
\end{tikzpicture}
```



`\Edges[`⟨*NotInBG*⟩`]{`*filename*`}`

Per default, the edges are drawn on the background layer of the *tikzpicture*. I.e. objects which are created after the edges appear also on top of them. To turn this off, the option ⟨*NotInBG*⟩ has to be enabled.

`\Edges[`⟨*vertices*⟩`=`*filename*`]{`*filename*`}`

Edges can be assigned to a specific set of `\Vertices` with the option ⟨*vertices*⟩. Thereby the argument *filename* is the same as used for the `\Vertices` command. This option might be necessary if multiple `\Vertices` are created and edges are assigned at the end.

`\Edges[`⟨*layer*⟩`={`*layer α*`,`*layer β*`}]{`*filename*`}`

With the option ⟨*layer*⟩ only the edges between layer α and β are plotted. The argument is a tuple of both layers indicated by { , }. More about this option and the use of layers is explained in Chapter 4.

4 Multilayer Networks

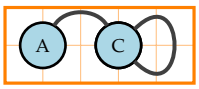
One of the main purposes of the `tkz-network` package is the illustration of multilayer network structures. Thereby, all the previous commands can be used. A multilayer network is represented as a three-dimensional object, where each layer is located at a different z plain. In order to enable this functionality, the option ⟨*multilayer*⟩ has to be used at the beginning of the `tikzpicture`.

4.1 Simple Networks

`\Vertex[`⟨*layer*⟩`=`*number*`]{`*Name*`}`

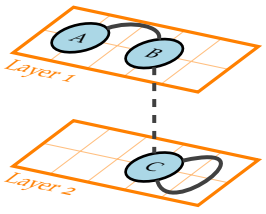
With the option ⟨*layer*⟩ the vertex can be assigned to a specific layer. Layers are defined by numbers (e.g. 1, 2, 3,...). Working with the ⟨*multilayer*⟩ option, each `\Vertex` has to be assigned to a specific layer. For the edge assignment no additional information is needed.

```
\begin{tikzpicture}[multilayer]
  \Vertex[x=0.5,IdAsLabel,layer=1]{A}
  \Vertex[x=1.5,IdAsLabel,layer=1]{B}
  \Vertex[x=1.5,IdAsLabel,layer=2]{C}
  \Edge[bend=60](A)(B)
  \Edge[style=dashed](B)(C)
  \Edge(C)(C)
\end{tikzpicture}
```



Enabling the option ⟨*multilayer*⟩, returns the network in a two-dimensional plane, like the networks discussed before. Setting the argument ⟨*multilayer*⟩=*3d*, the network is rendered in a three-dimensional representation. Per default, the layer with the lowest number is on the top. This and the spacing between the layers can be changed with the command `\SetLayerDistance`.

```
\begin{tikzpicture}[multilayer=3d]
  \Vertex[x=0.5,IdAsLabel,layer=1]{A}
  \Vertex[x=1.5,IdAsLabel,layer=1]{B}
  \Vertex[x=1.5,IdAsLabel,layer=2]{C}
  \Edge[bend=60](A)(B)
  \Edge[style=dashed](B)(C)
  \Edge(C)(C)
\end{tikzpicture}
```



4.2 *Complex Networks*

Similar as in Chapter 3 introduced, layers can be assigned to the vertices by adding a column ⟨*layer*⟩ to the file where the vertices are stored.

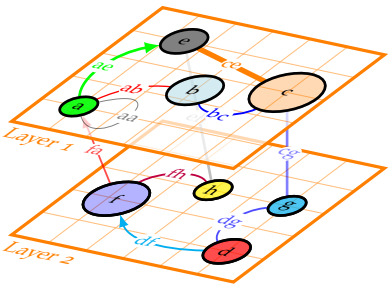
File: ml_vertices.csv

```
id, x, y ,size, color,opacity,label,layer
 A, 0,  0, .4 , green,  .9   ,  a  ,  1
 B, 1, .7, .6 ,      ,  .5   ,  b  ,  1
 C, 2,  1, .8 ,orange,  .3   ,  c  ,  1
 D, 2,  0, .5 ,   red,  .7   ,  d  ,  2
 E,.2,1.5, .5 ,  gray,       ,  e  ,  1
 F,.1, .5, .7 ,  blue,  .3   ,  f  ,  2
 G, 2,  1, .4 ,  cyan,  .7   ,  g  ,  2
 H, 1,  1, .4 ,yellow,  .7   ,  h  ,  2
```

File: ml_edges.csv

```
u,v,label,lw,color ,opacity,bend,Direct
A,B, ab  ,.5,red   ,   1   ,  30,false
B,C, bc  ,.7,blue  ,   1   , -60,false
A,E, ae  , 1,green ,   1   ,  45,true
C,E, ce  , 2,orange,   1   ,   0,false
A,A, aa  ,.3,black ,  .5   ,  75,false
C,G, cg  , 1,blue  ,  .5   ,   0,false
E,H, eh  , 1,gray  ,  .5   ,   0,false
F,A, fa  ,.7,red   ,  .7   ,   0,true
D,F, df  ,.7,cyan  ,   1   ,  30,true
F,H, fh  ,.7,purple,   1   ,  60,false
D,G, dg  ,.7,blue  ,  .7   ,  60,false
```

With the commands `\Vertices` and `\Edges`, the network can be created automatically. Again the `\Vertices` vertices should be performed first and then the command `\Edges`.

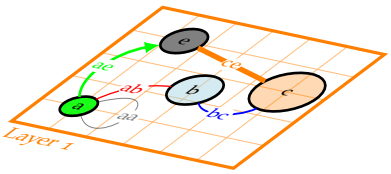



```
\begin{tikzpicture}[multilayer=3d]
  \Vertices{data/ml_vertices.csv}
  \Edges{data/ml_edges.csv}
\end{tikzpicture}
```

`\Vertices[`⟨*layer*⟩`=`*number*`]{`*filename*`}`

`\Edges[`⟨*layer*⟩`={`*layer* α`,`*layer* β`}]{`*filename*`}`

With the `\Vertices` option ⟨*layer*⟩ only the vertices on the selected layer are plotted. While, with the `\Edges` option ⟨*layer*⟩, the edges between layer α and β are plotted. The argument is a tuple of both layers indicated by { , }.




```
\begin{tikzpicture}[multilayer=3d]
  \Vertices[layer=1]{data/ml_vertices.csv}
  \Edges[layer={1,1}]{data/ml_edges.csv}
\end{tikzpicture}
```

Plotting edges without defining first the vertices can be done with the `\Edges` option ⟨*vertices*⟩. This allows modifying specific sets of Edges.

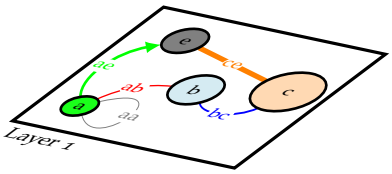
```
\begin{tikzpicture}[multilayer=3d]
  \Edges[vertices=data/ml_vertices.csv,
         layer={1,2},style=dashed]{data/ml_edges.csv}
\end{tikzpicture}
```

4.3 Layers and Layouts

Besides adding vertices and edges to specific layers, every other TikZ object can be drawn on such a layer using the `Layer` environment. With the option ⟨*layer*⟩=*layer α*, the position of the canvas can be assigned to the specific layer.

`\begin{Layer}[`⟨*layer*⟩=*layer α*`]`
`\end{Layer}`

```
\begin{tikzpicture}[multilayer=3d]
  \begin{Layer}[layer=1]
    \draw[very thick] (-.5,-.5) rectangle (2.5,2);
    \node at (-.5,-.5)[below right]{Layer 1};
  \end{Layer}
  \Vertices[layer=1]{data/ml_vertices.csv}
  \Edges[layer={1,1}]{data/ml_edges.csv}
\end{tikzpicture}
```



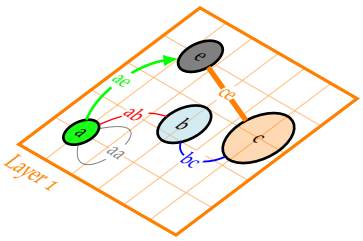
`\SetLayerDistance{`*measure*`}`

With the command `\SetLayerDistance` the distance between the layers and their orientation can be modified. Per default the distance is set to −2`\DefaultUnit` (here cm). A negative number implies that layers with a higher number will be stacked below layers with a smaller number.

`\SetCoordinates[`⟨*xAngle*⟩=*number*`,`⟨*yAngle*⟩=*number*`,`⟨*zAngle*⟩=*number*`,`
⟨*xLength*⟩=*number*`,`⟨*yLength*⟩=*number*`,`⟨*zLength*⟩=*number*`]`

The perspective of the three-dimensional plot can be modified by changing the orientation of the coordinate system, which is done with the command `\SetCoordinates`. Here the angle and the length of each axis can be modified. Angles are defined as a *number* in the range between −360 and 360. Per default, the lengths of the axes are defined by the identity matrix, i.e. no distortion. If the length ratio is changed *x*, *y*, and/or *z* values are distorted. The `\SetCoordinates` command has to be entered before the ⟨*multilayer*⟩ option is called!

```
\SetCoordinates[xAngle=-30,yLength=1.2,xLength=.8]
\begin{tikzpicture}[multilayer=3d]
  \Vertices[layer=1]{data/ml_vertices.csv}
  \Edges[layer={1,1}]{data/ml_edges.csv}
\end{tikzpicture}
```



5 Default Settings

In order to customize the look of the networks, each layout setting used can be modified and adapted. There are three categories: General settings, vertex style, and edge style.

5.1 General Settings

With the general settings mainly the sizes, distances and measures of the networks can be modified.

`\SetDefaultUnit{`*unit*`}`

The command `\SetDefaultUnit` allows to change the units used for drawing the network[1], including diameters of the vertices, x and y coordinates or the distance between the layers. The default unit is cm.

[1] Except the line width, which are defined in pt.

`\SetDistanceScale{`*number*`}`

With the command `\SetDistanceScale`, the distance between the vertices can be scaled. Per default 1 cm entered corresponds to 1 cm drawn, i.e. `\SetDistanceScale{1}`. Decreasing or increasing the scale changes the drawing distances between the vertices.

`\SetLayerDistance{`*measure*`}`

With the command `\SetLayerDistance` the distance between the layers and their orientation can be modified. Per default, the distance is set to -2. A negative number implies that layers with a higher number will be stacked below layers with a smaller number.

`\SetCoordinates[`⟨*xAngle*⟩`=`*number*`,`⟨*yAngle*⟩`=`*number*`,`⟨*zAngle*⟩`=`*number*`,`
⟨*xLength*⟩`=`*number*`,`⟨*yLength*⟩`=`*number*`,`⟨*zLength*⟩`=`*number*`]`

The perspective of the three-dimensional plot can be modified by changing the orientation of the coordinate system, which is done with the command `\SetCoordinates`. Here the angle and the length of each axis can be modified. Angles are defined as a *number* in the range between -360 and 360. Per default, the length of the axes are defined by the identity matrix, i.e. no distortion. If the length ratio is changed x, y, and/or z values are distorted. The `\SetCoordinates` command has to be entered before the ⟨*multilayer*⟩ option is called!

5.2 *Vertex Style*

The appearance of the vertices can be modified with the command `\SetVertexStyle`. This command will change the default settings of the vertices in the network.

`\SetVertexStyle{`*document options*`}`

The following options are available:

Option	Default	Type	Definition
VertexShape	circle	text	shape of the vertex
VertexInnerSep	2pt	measure	separation space which will be added inside the shape
VertexOuterSep	opt	measure	separation space outside the background path
VertexMinSize	0.6`\DefaultUnit`	measure	diameter (size) of the vertex
VertexFillColor	vertexfill	color	color of the vertex
VertexFillOpacity	1	number	opacity of the vertex
VertexLineWidth	1pt	measure	line width of the vertex boundary
VertexLineColor	black	color	line color of the vertex boundary
VertexLineOpacity	1	number	line opacity of the vertex boundary
VertexTextFont	`\scriptsize`	fontsize	font size of the vertex label
VertexTextColor	black	color	color of the vertex label
VertexTextOpacity	1	number	opacity of the vertex label
VertexTextRotation	0	number	initial rotation of the vertex

Table 5.1: Document style options for the vertices.

5.3 *Edge Style*

The appearance of the edges can be modified with the command `\SetEdgeStyle`. This command will change the default settings of the edges in the network.

`\SetEdgeStyle{`*document options*`}`

The following options are available:

Option	Default	Type	Definition
EdgeLineWidth	1.5pt	measure	width of the edge
EdgeColor	black!75	color	color of the edge
EdgeOpacity	1	number	opacity of the edge
EdgeArrow	-latex	text	arrow shape of the directed edge
EdgeTextFont	`\scriptsize`	fontsize	font size of the edge label
EdgeTextOpacity	1	number	opacity of the edge label
EdgeTextFillColor	white	color	fill color of the edge label
EdgeTextFillOpacity	1	number	fill opacity of the edge label
EdgeInnerSep	opt	measure	separation space which will be added inside the shape
EdgeOuterSep	1pt	measure	separation space outside the background path
EdgeTextRotation	0	number	initial rotation of the edge label

Table 5.2: Document style options for the edges.

6 Troubleshooting and Support

6.1 Tufte-LaTeX Website

The website for the `tkz-network` packages is located at https://github.com/hackl/tkz-network. There, you'll find the actual version of the source code, a bug tracker, and the documentation.

6.2 Getting Help

If you've encountered a problem with one of the `tkz-network` commands, have a question, or would like to report a bug, please send an email to me or visit our website.

To help me troubleshoot the problem more quickly, please try to compile your document using the `debug` class option and send the generated `.log` file to the mailing list with a brief description of the problem.

6.3 Errors, Warnings, and Informational Messages

The following is a list of all of the errors, warnings, and other messages generated by the `tkz-network` classes and a brief description of their meanings.

```
Error:  !  TeX capacity exceeded, sorry [main memory size=5000000].
```

The considered network is to large and `pdflatex` runs out of memory. This problem can be solved by using `lualatex` or `xetex` instead.

6.4 Package Dependencies

The following is a list of packages that the `tkz-network` package rely upon. Packages marked with an asterisk are optional.

- etex
- xifthen
- xkeyval
- datatool
- tikz
  - arrows
  - positioning
  - 3d
  - fit
  - calc
  - backgrounds

A ToDo

A.1 Code to fix

- change default entries for Boolean options in the vertices file.

A.2 Documentation

- add indices to the manual.
- add an extended tutorial/example to the document.
- clean-up and document the .sty file.
- upload the package to CTAN, if it is appropriated tested.

A.3 Features

- add `\Plain` command in order to create automatically multilayer plains.
- add a spherical coordinate system

A.4 Add-ons

- add igraph to tkz-network compiler (e.g. plot function)
- add networkx to tkz-network compiler (e.g. plot function)
- add QGIS to tkz-network compiler