CLDrawingEngine 1.0

White Paper

July 2006

CL Solutions Doris Chu & Thomas Langhagel GbR

info@cl-solutions.de

www.cl-solutions.de

CL Solutions
Doris Chu & Thomas Langhagel GbR

CLDrawingEngine 1.0
White Paper

CLDrawingEngine 1.0
White Paper

1. General

CLDrawingEngine is a Java library which can be used to easily and quickly develop powerful components for any kind of graphical visualisation and intuitive user interaction. The intention of this library is to provide the application developer with an easy-to-use programming interface, so that she can concentrate on the specific visualisation and user interaction requirements of the application, whereas the CLDrawingEngine takes over the task to do all the underlying necessary functionality to draw and handle interactive user requests. Using CLDrawingEngine development times for any kinds of model based complex visualisation tasks or graphical editor components can be dramatically reduced.

The CLDrawingEngine is optimised for handling complex drawings. So it is proven to handle thousands of nodes with ten-thousends of connections between them with user acceptable reaction times.

CLDrawingEngine is completely written in pure Java, so it is available for all relevant client platforms like Microsoft Windows, LINUX, UNIX and Apple. The deployment of CLDrawingEngine is done simply by deploying one JAR-file of less than 300 kByte in the applications classpath, so that it easily can be integrated in any standard installation procedure.

The functionality and features of the CLDrawingEngine are based on more than 15 years of experience in the development of interactive graphic applications, so the developer using CLDrawingEngine benefits from huge feedback from application users.

2. Developing Visualisation Components with CLDrawingEngine

2.1. Software Prerequisits

For the development of components based on CLDrawingEngine you need a valid license for the library. CLDrawingEngine is developed with Java JDK 1.5, so it is mandatory, that you use a Java Runtime Environment, which is compatible to Java 5.

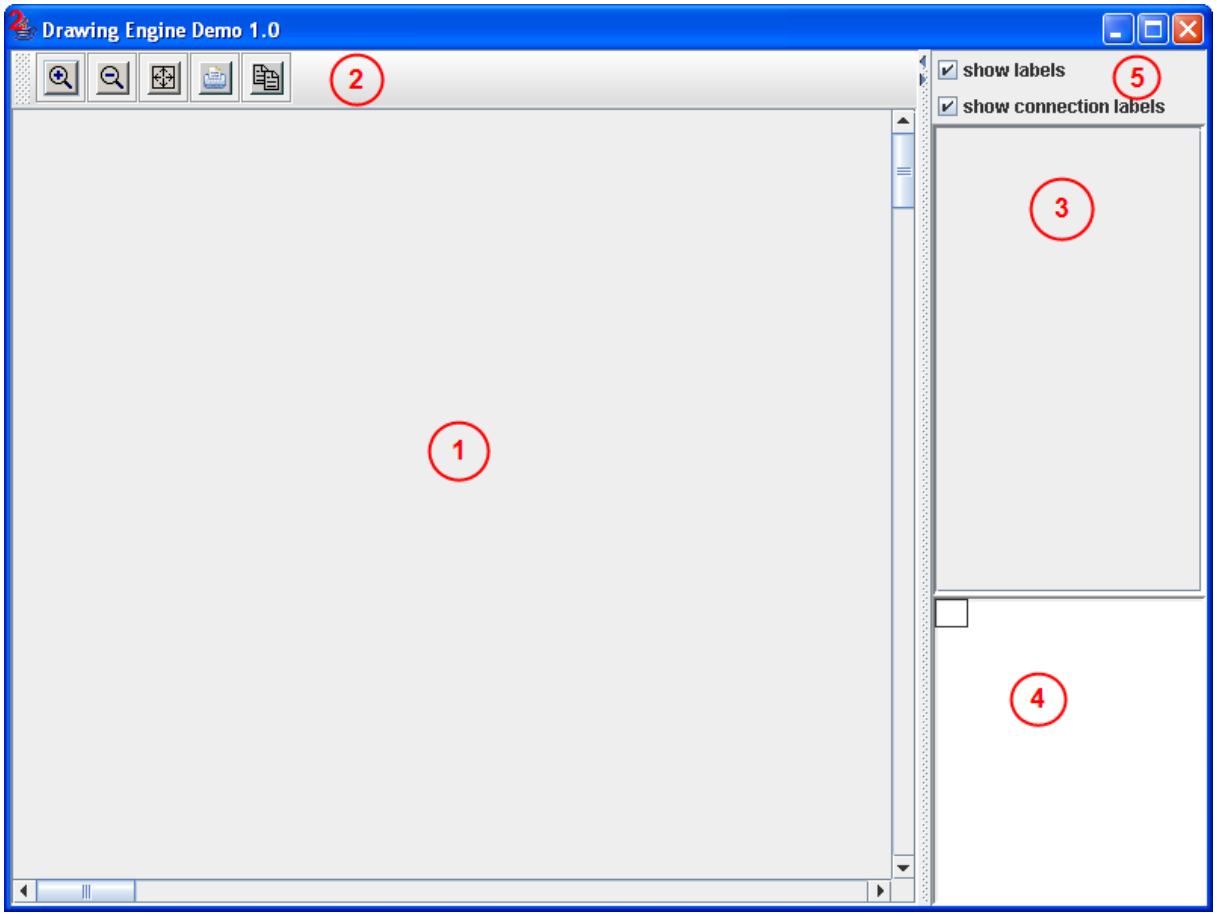
2.2. The Drawing Engine Panel

The core element of the CLDrawingEngine is the Drawing Engine Panel. It is a subclass of javax.swing.JPanel, so it can easily be integrated in any Java Swing application.
It is possible to use multiple independent instances of a Drawing Engine Panel within one application, each with its own set of registered handlers, drawing styles etc. So the CLDrawingEngine is best prepared for the use within complex graphic and visualisation driven applications.

CLDrawingEngine 1.0
White Paper

2.2.1. Components of the DrawingEnginePanel

The component DrawingEnginePanel consists of a number of different components as it is shown in the picture below:



	Component	Description
1	Drawing area	Drawing area, in which the drawing based on the loaded drawing model is shown.
2	toolbar	The toolbar contains controls for zooming, fitting, printing and copying to the clipboard. The CLDrawingEngine provides functionality to add own controls for specific functionality.
3	Layer Control Panel	The layer control panel lists all layers of the loaded drawing model. By checking/unchecking it is possible to hide/unhide specific layers.
4	Navigation Panel	The navigation allows to easily navigate from a bird perspective through large extensive drawings.
5	Drawing options	The drawing options control contains checkboxes for switching on/off the labels of the drawing, which is helpful for complex drawing with a lot of details, if the user wants to get an overview. The CLDrawingEngine provides functionality to add own checkbox controls for specialized drawing filters.

2.2.2. Scales modes

The drawing engine provides two different modes for the scaling of the drawing, when zooming in respectively zooming out.

- ***SCALE_MODE_ALL***

  When this scale mode is set (the default scale mode for a DrawingModel), any contents of the drawing is scaled proportional dependent to the scale factor. I.e. distances between nodes are scaled as well as the dimensions of any drawing element are scaled with the same proportion.

- ***SCALE_MODE_COORDINATE***

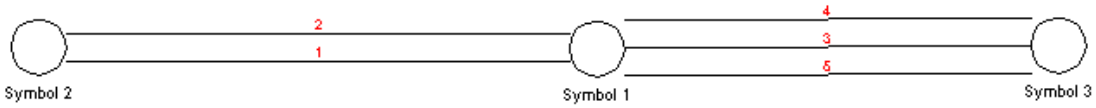
  When this scale mode is set, then only the dimensions between nodes are scaled dependent to the scale factor. I.e. the dimensions of any drawing elements (e.g. the radius of a circle) keep their dimensions.

2.2.3. Connection modes

The CLDrawingEngine provides two different modes to visualise connections between nodes:

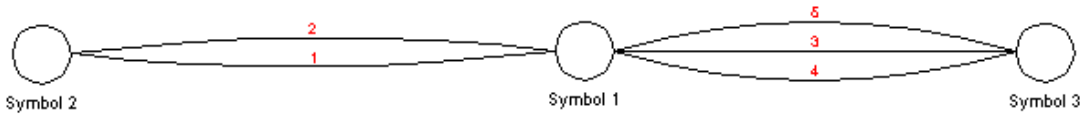
- ***Connection mode parallel***

When this connection mode is set, multiple connections between two identical nodes are drawn as straight direct lines, which are separated between each other by a defined offset.



- ***Connection mode curved***

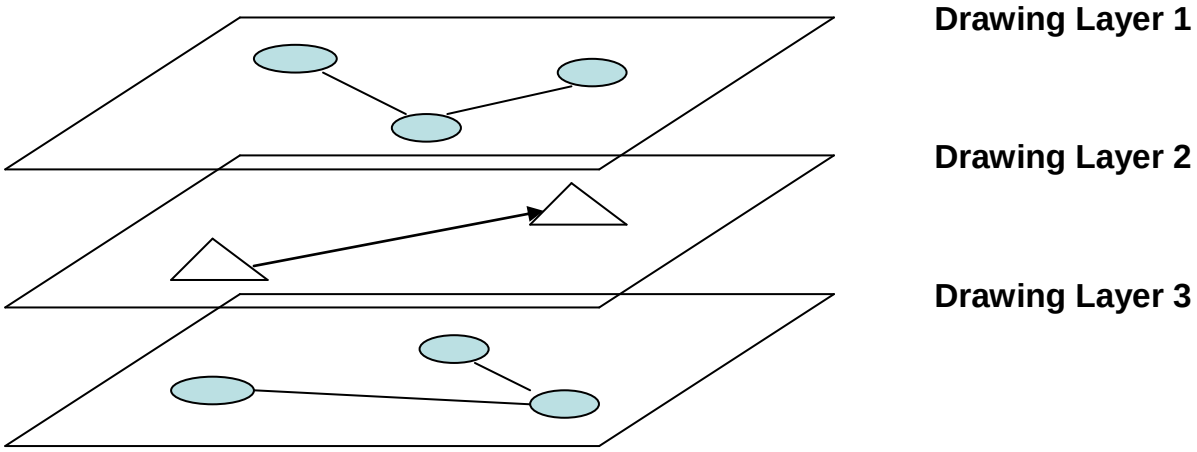
When this connection mode is set, multiple connections between two identical nodes are drawn as curved lines, which are separated between each other by a defined offset. This mode is preferable, when a larger number of connections between same nodes is apparent.



2.3. The Drawing Model

All information, which shall be visualised by the CLDrawingEngine has to be preparing in a drawing model.

A drawing consists of one or more drawing layers.



Each drawing layer can contain any number of drawing elements, which have to be instances of the interface **`IDrawingElement.`**

Drawing elements can be either nodes or connectors.

A node is the representation of a graphical item, which has at one time a specific position within the drawing. A node normally has a graphical representation in form of a symbol. The symbol can be generated e.g. by an image or can be a specialised drawing generated by a user defined subclass.

A connector is a visible connection between two nodes. So a connection cannot exist without a related start node and a related end node. The CLDrawingEngine allows connecting nodes, which are assigned to different drawing layer, with a connector, which can be assigned to its individual layer. Together with the layer control functionality, which is described later, this allows to temporarely visualise only specific aspects of a drawing by switching on/off specific layers. So for example it is possible to only show nodes and switch of the connectors, when these are assigned to separate drawing layers.

With the standard delivery of CLDrawingEngine come a number of predefined ready-to-use drawing elements, both nodes and connectors.

2.4. Standard Drawing Elements

2.4.1. Node elements

- ***Labeled Node***

A labeled node is an abstract base class for representing a node with an assigned text label. The label is drawn adjacent to the node. The relative position of the label can be controlled with a specific flag.

For a LabeledNode it is possible to specify the position of the label relative to the node symbol. This is controlled by the property “labelAlign”, which allows specifying the label alignment in a compass like manner.

With the method LabeledNode.flipLabelAlign() it is possible to stepwise change the position of the label. Stepping is done in clockwise direction.

- ***Symbol Node***

A SymbolNode is a specialised type of a LabeledNode. The graphical representation is done by means of an image. All standard image formats, which are supported by Java, can be used. The symbol images are managed by an intelligent resource pool, so only the minimum necessary system resources for loading and caching the images are used.

2.4.2. Connection elements

- ***Connector***

The Connector is the basic type for drawing connections between two nodes. The appearance of the Connector (colour, line thickness, line style) can be set by setting a specific DrawingStyle
When multiple Connector instances exist, which have the same pair of source and destination Node instances, the representation of the Connector depends on the connection mode for the drawing layer it is assigned to.

- ***Labelled Connector***

A labelled connector is based on the basic Connector. It additionally allows adding a text label, a start label and an end label, which are drawn in combination with the connector.

- ***Fork Connector***

A fork connector is a specialised connector type, which allows easily representing 1-to-many relations within a drawing. The individual connectors are drawn in a fork like manner.

2.4.3. Combining with the business layer

It is possible to assign to each individual drawing element any user object as Java Object. So this allows to carry any information about the business objects with the graphical representation in form of the drawing elements.
Example: a Connector representing a physical link in a telecommunications network can be assigned to a business object with all information about this link.
Any user interaction handlers like mouse double click handlers, which are described later, have access to these objects. Also the drawing style of such an object can be adjusted related to the state of the business object.
So it is possible to control the reaction on such interaction events dependent to the business logic.

2.5. Configurable Context menus

It is possible to register specific context menus with specific kinds of drawing elements. So specific functionality can be added, which allows the user to perform specific elements of the drawing. The application programmer only has to provide the specific actions as subclassed Action classes and register them class-specific with a specialised context menu. The CLDrawingEngine cares for all necessary internal handling, when the user clicks with the right mouse button on a related drawing element.

2.6. Configurable user interaction

Interacting with a drawing within the drawing panel can be done in several ways. It is possible to control the behaviour of mouse clicks, mouse dragging, tooltips etc. by registering specific handler. These handlers can be registered specifically to specific drawing element classes.
The following different types of handlers can be specified:

2.6.1. Mouse single click handler

This handler is invoked, when the user clicks one times on a drawing element, for which the handler is registered.
Example: for specific connectors the handler triggers the highlighting of other drawing elements, which are related to the object represented by the connector.

2.6.2. Mouse double click handler

This handler is invoked, when the user doube clicks on a drawing element, for which the handler is registered.

Example: When the user double clicks a specific node, the property editor for the represented business object is popped up.

2.6.3. Tooltip handler

The tooltip handler controls the contents of the tooltip for the drawing element, when the user moves the mouse cursor above the drawing element.
With the handler the application programmer can apply any logic to build up the tooltip dynamically.

2.6.4. Dragged element handler

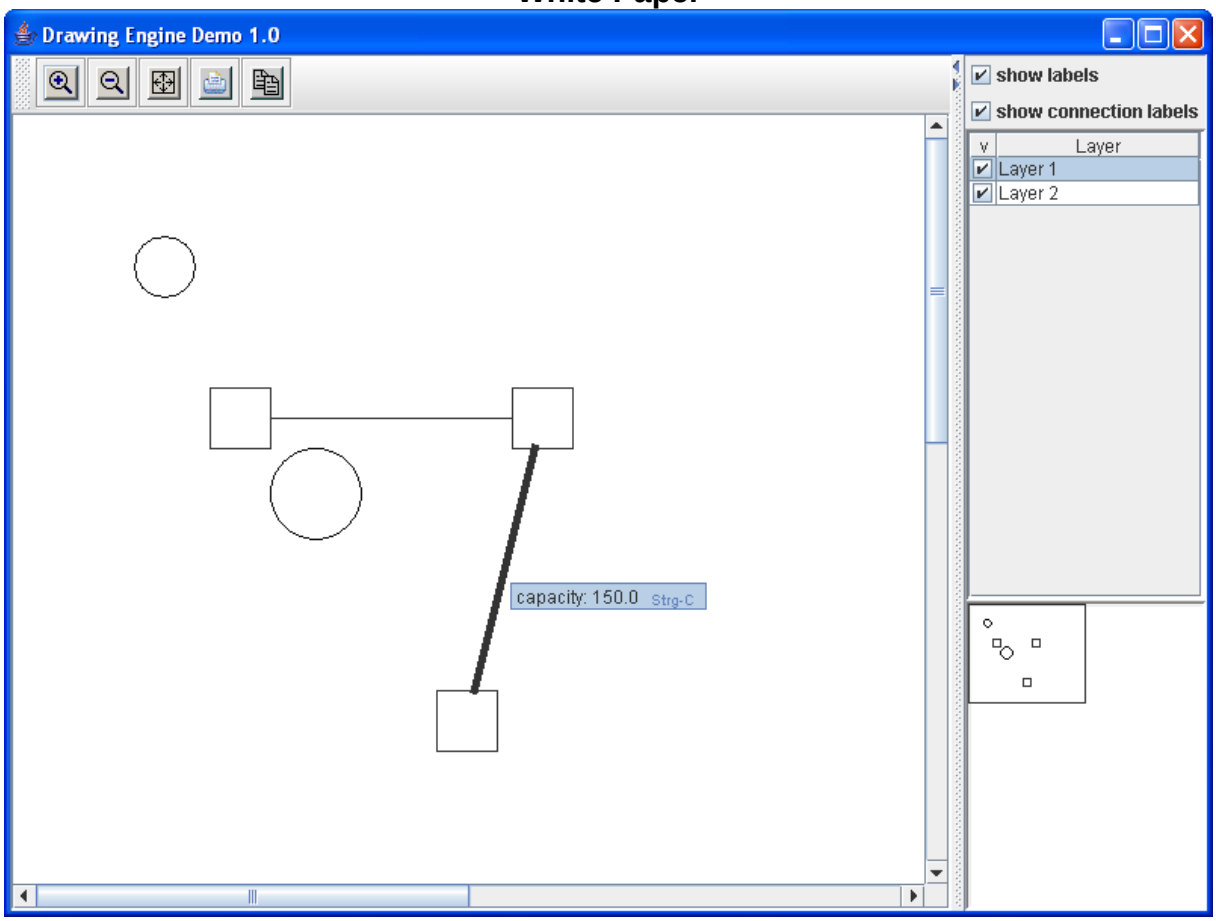
This handler allows to apply specific business logic for nodes, which are dragged by the mouse and dropped above another node. The handler is invoked automatically, when the drop occurs above an drawing element, for which the handler is registered.

2.6.5. New Connection handler

This handler is invoked, when the user presses the mouse button on a node of the drawing with at the same time pressed <ALT> key. This starts the creating of a new connection between nodes by drag&drop. The user gets a visual feedback by means of a preview line, which is drawn from the source object, until the mouse is released above the destination node. The application programmer can provide any logic within the handler, which is required for this user interaction in terms of the business logic.

2.6.6. DrawingStyle handler

With a specific drawing style handler registered for a specific drawing element class it is possible to dynamically change the appearance of a drawing element.
Example: The line thickness and color of connectors representing a physical link are set proportional to the capacity and load of the represented link object.



2.7. Drawing Styles

Any drawing element is drawn by default with standard drawing attributes, i.e. line colour black, standard font, standard line thickness, standard line pattern etc.
To modify this behaviour it is possible to assign to each drawing element a specific DrawingStyle object. This can be done both for a complete drawing element class and for individual drawing elements.
Because a drawing style, if set, is evaluated during repainting, it is also possible to specify a dynamic behaviour of the drawing style, e.g. related to the state of a specific drawing element.
Example: the line thickness of a connector representing a communication link is set related to the capacity of the link.

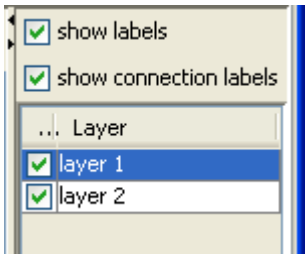
2.8. Drawing Options

Drawing options allow controlling specific behaviour of the drawing. E.g. you can use a drawing option to switch on or off a specific drawing style for a set of elements like coloring or line thickness. In the user interface of the drawing panel drawing options are represented by checkboxes, so that the user can switch on or off an option by checking respectively

unchecking the checkbox. Each drawing option checkbox is constructed by a specific action, which is invoked when the drawing option's state is toggled.
The application programmer can add easily own drawing options by simply registering an Action object, which handles the required behaviour.

2.9. Layer Management

The drawings visualised with the CLDrawingEngine are organised in drawing layers. This allows to combine related visualisation aspects in such layers. The user can switch on or off specific layers, if she wants to focus on a specific aspects of the drawing. This is most important in complex drawings, where several different aspects of visualisation are combined in one drawing model.
It is also possible to specify dependencies between layers, i.e. one or several layers are only shown, if the layer, on which they depend on is shown. This allows to build up a drawing model with fine grained structurisation.



3. Event Handling

The CLDrawingEngine has a built-in standard signalling mechanism, which allows informing any interested listeners about modifications or selections which are made within the drawing. So it is possible to for an application using the CLDrawingEngine to react on the user interaction within a drawing.
Also th CLDrawingEngine provides functionality, which allows the drawing to react on events from the containing application. So for example it is possible to force a reload of the drawing, when the business layer, which is represented by the drawing, has changed.

4. Printing

The CLDrawingEngine has built-in functionality, which allows printing out any contents of a drawing visualised in a DrawingPanel. Optionally it is possible to add a configurable frame around the drawing with information like title, date and author of the document.
The printing functionality is based on the standard Java functionality, so any printer connected to the computer can be used.

5. Clipboard Functionality

With the built-in clipboard functionality a user can easily make a copy of the current drawing and paste it into any external document, which supports the underlying clipboard functionality of the operating system.

6. Contact

If you are interested in more information about the CLDrawingEngine or any project support we can help you in this area, please contact us:

CL Solutions
Doris Chu & Thomas Langhagel GbR

Volksdorfer Weg 2a
D-22391 Hamburg
www.cl-solutions.de

Phone: +49-40-636649-03
eMail: info@cl-solutions.de