Date: 25-10-2001
Subject: ModelViewer Documentation
Author: Sander Groenen

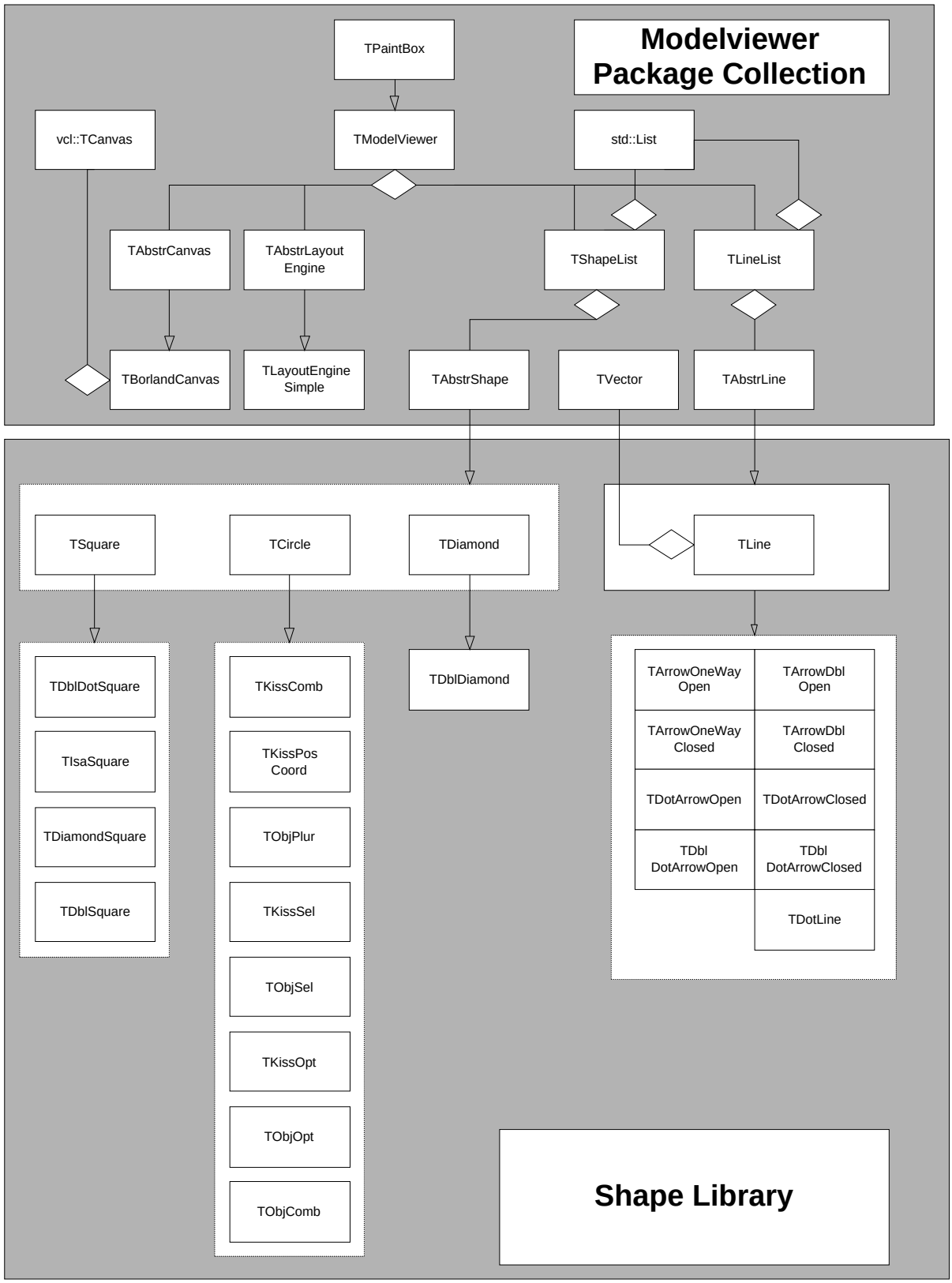
InhoudsOpgave

1. ModelViewer Technical Documentation

1.2 Introduction

The modelviewer component was constructed to be able to view all kinds of models that are created with the modelbuilder program. With the modelviewer one can put together any kind of model that is used within the kiss modelling language, like an object diagram or a kiss model. Basically the modelviewer consists of two elemental parts, namely the modelviewer package collection, which allows you to draw shapes on the canvas, search through your shapes and lines or set properties such as linewidth, color or penstyle, and second the shape library which consists of all the available shapes to use in the modelviewer such as a double dotted square or an kiss position coordinator symbol.
The figure in paragraph 1.3 below shows all separate components of the modelviewer package collection and the shape library and it also shows what kind of hierarchy is used. For more essential information on the individual components see chapter 2 and 3.

1.3 Modelviewer Design



2 ModelViewer Package Collection explanation

2.1 Modelviewer

The modelviewer component is a composite that exists of a TabstrCanvas, a TshapeList, a TlineList, a TabstrLayoutEngine. It provides methods to draw all shapes, set line width , set filling color etc. In other words its methods, like any composite does, are written in terms of it's composites. The only individual functionality it provides are events to respond to, such as an onpaint event handler or onmousemove event handler. These handlers are used to draw all the shapes and lines on the canvas and to drag a shape across the canvas.

2.2 TabstrCanvas

The TabstrCanvas is a pure virtual abstract base class for a canvas class and it provides methods for drawing on a canvas such as lineto, circle, ellipse, text rectangle etc. Pure virtual means that the methods it provides are only declared but **not** defined. It only provides a base class from which other none pure virtual canvas classes can be derived and thereby defining the declared methods.

2.2.1 TBorlandCanvas

The TBorlandCanvas class is derived from the TabstrCanvas described above and implements all the methods declared in TabstrCanvas using specific code for a Tcanvas class of borland that resides in the borland main library **vcl.**

2.3 TabstrLayoutEngine

The TabstrLayout engine is a pure virtual abstract base class for a layout engine class and it provides method to rearrange the co-ordinates of al l currently present shapes so none of the shapes will overlap each other on the canvas. Again it only has methods that are declared but **not** defined.

2.3.1 TlayoutEngineSimple

The TlayoutEngineSimple class is derived from the TabstrLayoutEngine class and implements all the methods declared in TabstrLayoutEngine using a simple algorithm for placing all the shapes, namely: put all shapes under each other in the middle of the canvas, regardless of any interconnection by means of lines.

2.4 TshapeList and TlineList

The two lists in the modelviewer component are used to store, add and delete all shapes and lines within the modelviewer. Both lists are composites because they both contain the template list type of the standard C++ library: **std.** Although slightly different, the two lists show great comparison; they both provide methods to go through, search, delete and add the elements(shapes,lines) .

2.4.1 TabstrShape and TabstrLine

TabstrShape and TabstrLine are two abstract base classes that are not pure virtual. They provide a basic functionality that all shapes and lines should have, such as setting their co-ordinates, width and height. All shapes derived from these classes extend their functionality by defining how they are drawn , the rest they retrieve from these two base classes.

3 Shape library explanation

3.1 Tsquare, Tcircle and Tdiamond

Tsquare, Tcircle and Tdiamond are three classes, that serve as bases classes for all other shapes. They implement their draw functionality by making, as obvious as would be, a square a circle and a diamond shape. All other Shapes use this functionality to draw their own shape.

3.2 Derived From Tsquare

When drawing, all shapes derived from Tsquare first use the draw method of Tsquare before they draw their "extras". For example a Double Dotted Square (TDblDotSquare) first draws a Square using Tsquare and then draws a dotted square within that one.

3.3 Derived From Tcircle

When drawing, all shapes derived from TCircle first use the draw method of TCircle before they draw their "extras". For example an object combination symbol (TObjComb) first draws a circle using TCircle and then draws a diagonal cross within that circle.

3.4 Derived From Tdiamond

When drawing, all shapes derived from TDiamond first use the draw method of TDiamond before they draw their "extras". For example a double diamond(TDblDiamond) first draws a Diamond using TDiamond and then draws another diamond within the first one

3.5 Derived from TLine

Currently there are ten types of arrow classes, namely a simple line with no arrowheads, a simple dotted line with no arrowheads,a one way open arrow(head), a one way closed arrow(head), a one way double open arrow(head), a one way double closed arrow(head), a one way dotted open arrow (head), a one way closed dotted arrow(head), a one way open dotted double arrow (head) and a one way dotted closed double arrow (head) .They are all derived from Tline(which in turn is derived from TAbstrLine) and differ only in number and shape(open or closed) of arrowheads or pen style(dotted or not dotted).

3.6 Tvector

Tvector class is a class, which resides in the modelviewer package collection, but is intended to be used with shapes. The class implements all standard 2 dimensional vector methods. These methods can be and are used for calculating interconnection between shape-edge points. The reason for putting this class in the package is the following: A user which does’nt have the shape library but wants to construct his own figures only needs the modelviewer package, because he has al the tools he needs, namely TabstrLine, TabstrShape and Tvector classes.