Manual for wxGraphLayout 1.0: a graph layout library for wxWindows

Julian Smart
Artificial Intelligence Applications Institute
University of Edinburgh
EH1 1HN

November 1993

Contents

Copyright notice

1. Introduction

This manual describes a graph layout class library for wxWindows. It is an encapsulation of the main code from Jos van Eijndhoven's graphplace tool which he has distributed to the net.

Below is the example graph generated by the program test.cc.

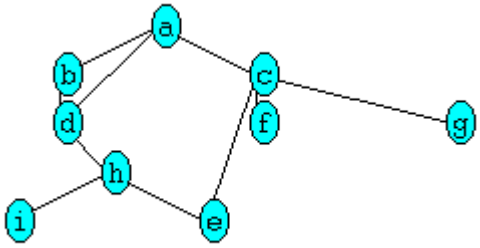

Figure 1: Example graph

Sorry, the documentation for wxGraphLayout is sparse at present, but see the class reference and the test program. Here is the message from Jos introducing his original graphplace tool:

We are using in our group a locally developed graph placement and drawing program, which was noticed as being very nice by several visitors. The program was made as general applicable tool, and indeed quickly gained use in several very different applications. Therefore we decided to make it available to all of you...

In general it will try to find a placement with 'short' edges, and in a 'breath-first-search' (sorted on level) order. The program works for any graph (also cyclic or not fully connected). The placement algorithm is a very fast linear-time heuristic, which often gives surprisingly nice results.

If you are interested, you can fetch the program from 'ftp.es.ele.tue.nl' as 'pub/down/graphplace.tar.Z'.

2. wxGraphLayout Class Reference

The member functions are given in alphabetical order except for the constructors and destructors which appear first.

2.1. wxGraphLayout: wxObject

This abstract class is used for drawing a graph. You don't have to derive a new class, but if you do provide *SetNodeX* (page 6) and *SetNodeY* (page 6) members, these will automatically be called to position your nodes. If you do not derive a new class and override these members, you need to call *GetNodeX* (page 4) and *GetNodeY* (page 4) for each node after the call to *DoLayout* (page 3).

Nodes are identified by long integer identifiers. The application should call *AddNode* (page 2) and *AddArc* (page 2) to register the nodes and arcs with wxGraphLayout, before calling *DoLayout* to do the graph layout. Depending on how the derived class has been defined, either *wxGraphLayout::Draw* must be called (for example by the OnDraw member of a wxCanvas) or the application-defined drawing code should be called as normal.

For example, if you have an image drawing system already defined, you may want wxGraphLayout to position existing node images in that system. So you just need a way for wxGraphLayout to set the node image positions according to the layout algorithm, and the rest will be done by your own image drawing system.

wxGraphLayout::wxGraphLayout

void wxGraphLayout(**wxDC ***_dc_ = _NULL_)

Constructor. *dc* is an optional device context for the class to draw the graph into.

wxGraphLayout::ActivateNode

void ActivateNode(**long** _id_, **Bool** _active_)

Call this to turn off nodes in the graph (not implemented yet). See also *NodeActive*.

wxGraphLayout::AddArc

void AddArc(**long** _id_, **long** _fromId_, **long** _toId_,**char ***_name_ = _NULL_)

Call this to add an arc to the graph, with optional name to display.

wxGraphLayout::AddNode

void AddNode(**long** _id_, **char ***_name_ = _NULL_)

Call this to add a node to the graph, with optional name to display.

wxGraphLayout::Clear

void Clear(**void**)

Clears the graph so another graph may be defined and laid out.

wxGraphLayout::DoLayout

void DoLayout(**void**)

Calculates the layout for the graph.

wxGraphLayout::Draw

void Draw(**void**)

Call this to let wxGraphLayout draw the graph itself, once the layout has been calculated with *DoLayout*. The device context must have been set in the constructor or using *SetDC*.

wxGraphLayout::DrawArc

void DrawArc(**long** *from*, **long** *to*)

Defined by wxGraphLayout to draw an arc between two nodes.

wxGraphLayout::DrawArcs

void DrawArcs(**void**)

Defined by wxGraphLayout to draw the arcs between nodes.

wxGraphLayout::DrawNode

void DrawNode(**long** *id*)

Defined by wxGraphLayout to draw a node.

wxGraphLayout::DrawNodes

void DrawNodes(**void**)

Defined by wxGraphLayout to draw the nodes.

wxGraphLayout::GetDC

long GetDC(**void**)

Gets the (optional) device context associated with the graph.

wxGraphLayout::GetNextNode

long GetNextNode(**long** *id*)

Must be defined to return the next node after *id*, so that wxGraphLayout can iterate through all relevant nodes. The ordering is not important. The function should return -1 if there are no more nodes.

wxGraphLayout::GetNodeSize

void GetNodeSize(**long** *id*, **float** **x*, **float** **y*)

Can be defined to indicate a node's size, or left to wxGraphLayout to use the name as an indication of size.

wxGraphLayout::GetNodeX

float GetNodeX(**long** *id*)

Must be defined to return the current X position of the node. Note that coordinates are assumed to be at the top-left of the node so some conversion may be necessary for your application.

wxGraphLayout::GetNodeY

float GetNodeY(**long** *id*)

Must be defined to return the current Y position of the node. Note that coordinates are assumed to be at the top-left of the node so some conversion may be necessary for your application.

wxGraphLayout::GetLeftMargin

float GetLeftMargin(**void**)

Gets the left margin set with *SetMargins*.

wxGraphLayout::GetRotation

int GetRotation(**void**)

Get the rotation factor.

wxGraphLayout::GetTopMargin

float GetTopMargin(**void**)

Gets the top margin set with *SetMargins*.

wxGraphLayout::GetXSpacing

float GetXSpacing(**void**)

Gets the horizontal spacing between nodes.

wxGraphLayout::GetYSpacing

float GetYSpacing(**void**)

Gets the vertical spacing between nodes.

wxGraphLayout::Initialize

void Initialize(**void**)

Initializes wxGraphLayout. Call from application or overridden **Initialize**or constructor.

wxGraphLayout::CalcLayout

void CalcLayout(**long** *node_id*, **int** *level*)

Private function for laying out a branch.

wxGraphLayout::NodeActive

Bool NodeActive(**long** *id*)

Define this so wxGraphLayout can know which nodes are to be drawn (not all nodes may be connected in the graph). See also *ActivateNode*.

wxGraphLayout::SetBoundingBox

void SetBoundingBox(**float** *x1*, **float** *y1*, **float** *x2*, **float** *y2*)

Sets the size of the bounding box to which the graph will be scaled. Pass the top left corner and bottom right corner.

wxGraphLayout::SetDC

void SetDC(**wxDC ***_dc_)

Use this to set the graph's device context, if leaving the drawing up to wxGraphLayout.

wxGraphLayout::SetNodeName

void SetNodeName(**long** *id*, **char *** *name*)

May optionally be defined to set a node's name.

wxGraphLayout::SetNodeX

void SetNodeX(**long** *id*, **float** *x*)

Must be defined to set the current X position of the node. Note that coordinates are assumed to be at the top-left of the node so some conversion may be necessary for your application.

wxGraphLayout::SetNodeY

void SetNodeY(**long** *id*, **float** *y*)

Must be defined to set the current Y position of the node. Note that coordinates are assumed to be at the top-left of the node so some conversion may be necessary for your application.

wxGraphLayout::SetRotation

void SetRotation(**int** *rot*)

Set the rotation factor (multipled by 90 degrees by wxGraphLayout).

wxGraphLayout::SetSpacing

void SetSpacing(**float** *x*, **float** *y*)

Sets the horizontal and vertical spacing between nodes in the graph.

wxGraphLayout::SetMargins

void SetMargins(**float** *x*, **float** *y*)

Sets the left and top margins of the whole graph.