

3D++ Class Library

Revision 1.0

Windows NT

Windows 95

Compiler Requirements - Visual C++ 4.2, 5.0 used with MFC.

Operating System Requirements - Windows NT or Windows 95. OpenGL may need to be installed on target systems which use Windows 95.

Simple Chart Objects - Line, Ribbon, Bar, Area Fill, Range Fill, Scatter Symbol, Scatter Text, Plane.

Group Chart Objects - Stacked Bar, Stacked Ribbon, Stacked Fill, Group Bar.

Compound Chart Objects - Open-High-Low-Close, Box and Whisker.

Polysurface Chart Objects - 3D Polysurface, 2D (Projection) Polysurface, Contour Lines and Surfaces, Independent variable mapping.

Axes - Tick marks, log axes, walls, grids, arbitrary XY intercepts, 3D rotated axis labels, string labels, manual and auto-axes routines, multiple axes, multiple scalings.

Lighting - A light source positioned in 3D space can be used to illuminate a chart.

Serialization - All charts can be saved to a file using built in serialization.

Legends - Legends can be used as symbol and color keys for 3D graphs.

3D TrueType Text - Select any TrueType font as the font for a graph.

No program limits for:

- Number of data points in a graph
- Number of chart objects in a graph
- Number of axes in a graph
- Number of text objects in a graph
- Number of 3D graphs in a view
- Number of legends in a view

No Runtime Royalties

Printing and Print Preview Support

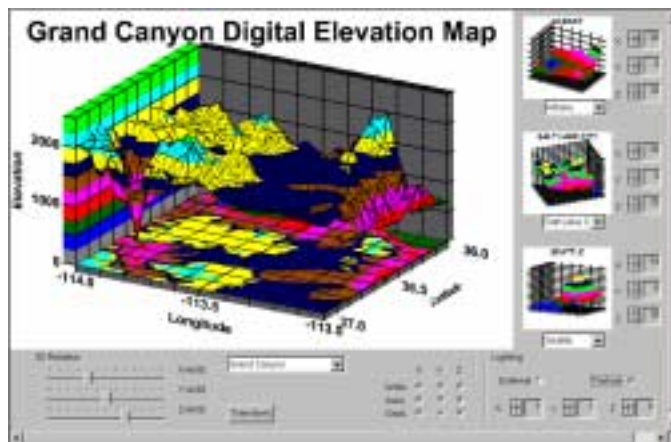
Windows Bitmap Support - Graphs can be saved as Windows bitmap files (DIBs). Windows DIBs can be loaded from a file, rotated in 3D and placed in a 3D graph.

Graph Enhancements - Add arbitrary text, legends, bitmaps, and geometric shapes to graphs.

Clipboard Support - 3D graphs can be copied to other applications using the clipboard.

Designed for MFC - Compatible with all CView and CDialog derived windows including CFormView and CScrollView.

Documentation and Example Programs - Includes a comprehensive user manual with many programming examples, on-line help and more than 20 complete demo programs (on disk).



Four separate 3D surface charts with contour texturing are combined with controls in a CFormView derived class.

The 3D++™ class library is written in C++ from the ground up using Visual C++™. 3D graphs can be placed in any MFC view, occupying the full view or any portion of a view. 3D graphs can also be combined with controls in dialogs based on CDialog and views based on CFormView. The library is implemented as a DLL which adds the 3D classes directly to your application program.

The 3D graphics engine used by the 3D++ class library is OpenGL, the 3D graphics library developed by Silicon Graphics, Inc. OpenGL is the foundation for many of the most advanced 3D imaging systems in the world. Originally developed for use in the Silicon Graphics line of graphics workstations, Microsoft licensed a version of OpenGL for use with Windows NT and Windows 95. OpenGL is a Microsoft "redistributable" and can be included in third party application programs without royalties.

The 3D++ class library is for use with Visual C++ 4.2 and higher and MFC. It includes a comprehensive manual that documents the 300+ member functions. On-line help and 20 example programs are also included.

Ordering Information

| Part # | Product Name | Price |
|-------------|--------------|-------|
| WIN-M3D-300 | 3D++ | \$500 |

SHIPPING CHARGES

| UPS Ground | UPS Blue | UPS Red | DHL | Canada |
|------------|----------|---------|-----|--------|
| 11 | 18 | 28 | 48 | 15 |

Product Description

WIN-M3D-300

3D++ Class Library for MFC

This product contains the 3D++ class library in DLL form, a user manual, on-line help and example programs. Revision 1.0 is only compatible with MFC based programs compiled using Visual C++ 4.2. The latest versions of OpenGL for both Windows NT and Windows 95 are included. The software is 32-bit compatible only and it does not run under Win32s. The source code for the 3D++ DLLs is not available.

System Requirements

The 3D++ class library is designed to run on Pentium class computers. While it will run on slower computers, rendering time will be proportionately slower.

- A computer configured for one of the following operating systems: Windows NT 3.51, 4.0 or Windows 95.
- A graphics board which supports a minimum of 256-colors.
- A complete installation of Visual C++ 4.2 and MFC.
- OpenGL DLLs, include files and libraries. The latest versions of these files can be downloaded from the Microsoft Web site at www.microsoft.com for free.

Programming with the 3D++ Class Libraries

Introduction

The 3D++ Class Library adds 3D charting and visualization capability to your MFC based programs. Simple XY data, normally plotted in 2D graphs, can make a much stronger statement when plotted as 3D area fill graphs, bar graphs or ribbon graphs. 3D data (XYZ) can be plotted as 3D scatter plots, lines or bars. The graph is easily rotated for the best possible view. Complex surfaces can also be graphed using advanced poly surface, contour and texture routines.

OpenGL

A decision was made in early 1996 to utilize the OpenGL 3D graphics engine as the basis for this software. OpenGL provides all of the 3D primitives used by 3D++ to draw 3D lines, polygons and text. OpenGL is an incredibly powerful software tool, but it is not easy to use by a typical Windows C programmer. It is difficult to get started writing MFC based OpenGL applications since all of the OpenGL examples Microsoft ships with Visual C++ are written in C, not MFC based C++.

Microsoft licenses OpenGL for Windows from Silicon Graphics, a company whose primary business is manufacturing advanced graphics workstations. OpenGL has also been licensed for use by other operating systems including UNIX. The initial release of OpenGL for Windows NT and Windows 95 was Revision 1.0. It ran reasonably fast under Windows NT but was 3-4 times slower under Windows 95. The current release of OpenGL for Windows NT and Windows 95 is Revision 1.1. This version has improved the

performance 2-4 times for Windows NT and up to 10 times for Windows 95. Since OpenGL is a well established standard, manufacturers of graphics boards are adding hardware accelerators to their products to speed up OpenGL rendering.

Revision 1.1 of OpenGL is included with all versions of Windows NT 4.0. Revision 1.1 of OpenGL for Windows 95 is expected to ship with Windows 95 in January 1997. You can get the latest version of OpenGL for Windows 95 or Windows NT by downloading it from the Microsoft software libraries at www.microsoft.com. From this web site search for OpenGL. Remember: the OpenGL Revision 1.1 DLLs for Windows NT and Windows 95 are different. They have the same name but different sizes. We have heard that at some time in the future the OpenGL DLLs will be unified, and you will not need different DLLs based on the operating system. This is not case as of January 1, 1997.

3D++ Basics

Data

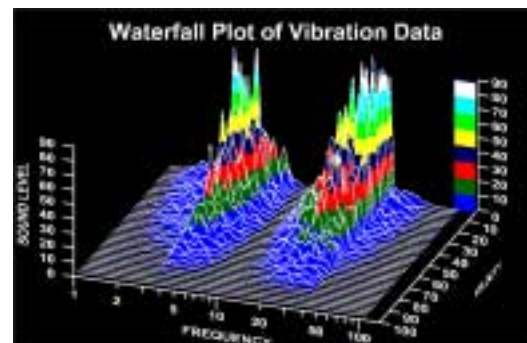
All of the numeric data plotting functions work with data organized in one of two ways: arrays of floating point numbers where each array holds one set of independent (X) or dependent (Y) data; or arrays of 3D points, where each 3D point is an element of type `point3Dtype`. The `point3Dtype` structure is defined as:

```
typedef struct {
    realtype x,y,z, v;
} point3Dtype;
```

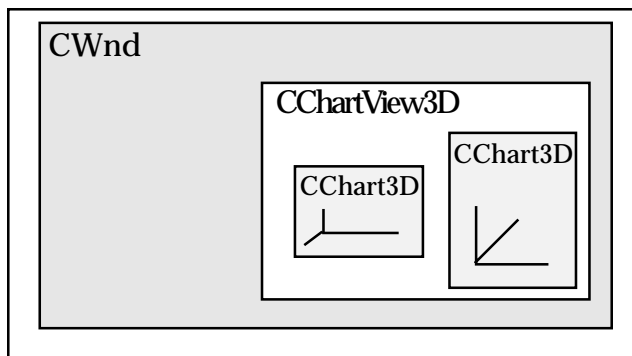
where `x`, `y` and `z` correspond to the associated X, Y or Z coordinate and `v` corresponds to a fourth independent value which can be assigned to each point. The `v` element is useful for 3D charts where you want texturing based on an independent variable, such as temperature, which has been measured at each 3D point. Data arrays should be allocated using the standard C++ `new` operator. An internal copy of all data passed to objects is made by the software.

Color & Lighting

The 3D++ class library requires a system with 256 (8-bit) colors minimum. This is an absolute OpenGL requirement. All colors are specified using RGB values. A structure named `RGBDType` is used to pass all color information to and from member functions. This is similar to the Win-



This waterfall plot consists of 32 `AreaFillChart25D` objects.



A *CView* object can contain one or more *CChartView3D* objects. Each *ChartView3D* object can contain one or more *CChart3D* objects.

uses RGBQUAD structure used by many Windows functions, except that RGBQUAD uses bytes in the range 0 to 255 to store RGB data and the RGBDType uses floating point numbers in the range 0.0 to 1.0.

In the absence of lighting or texturing, the color of the graphical object will be a solid, uniform color, the value of the object color attribute. If lighting is used, the color of the graphical object varies around its attribute color, depending on the angle of the object surface compared to the angle of the light. The more perpendicular the object is to the light, the lighter the color is. Object planes which are parallel to the light source are in shadow and hence dark.

Fonts and Text Attributes

In this first release of the 3D++ class library, only one OpenGL TrueType font is active at a time. The font that you select for a *CChartView3D* window is the font for all graphs in that window. Text printed in the window can have different sizes, but all text will be in the same font.

3D++ Class Summary

CChartView3D

The *CChartView3D* class contains one or more *CChart3D* objects. The *CChartView3D* object maps to the client area of the associated MFC view. The programmer can place *CChart3D* objects at any position inside the rectangle defined by *CChartView3D*. The *CChart3D* objects are stored in a linked list.

CChart3D

A *CChart3D* class is contained within a *CChartView3D* class. The *CChart3D* class contains one or more of the following classes:

| | |
|---------------------|---|
| CAxes3D | Axes, axes labels and axes titles. |
| CChartObj3D | Bar graphs, area fill graphs, surface charts, i.e. all of the data display routines for a 3D graph. |
| CTransform3D | Holds a set of scaling and rotation parameters. |
| CTextObj3D | Text object which can be positioned in 3D space. |

CDIBObj3D DIBs (device independent bitmaps) can be used to overlay arbitrary rectangular planes in a graph.

These classes are commonly referred to as charting objects because they are objects that reside inside the *CChart3D* class. All charting objects are stored in linked lists and there is no upper limit on the number of charting objects that can be added to a 3D graph. The *CChart3D* class has various attributes which can be set including two headers, a footer, and lighting parameters.

CTransform3D

A *CTransform3D* object defines a unique set of scaling and rotational parameters used when rendering 3D objects on the screen. Objects of this type are created by calling the *NewTransform* member function of a *CChart3D* object. Each time the *NewTransform* member function is called, a new *CTransform3D* object is created and added to the linked list maintained by the parent *CChart3D* object. All charting objects reference a *CTransform3D* object when created. This allows you to create multiple transforms, each with different scales, and create *CChartObj3D* objects which are rendered using the associated *CTransform3D* object.

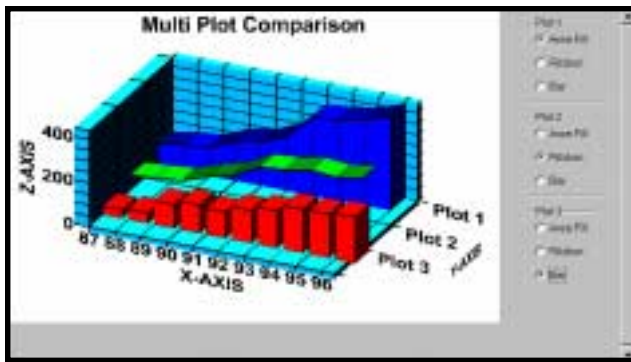
CAxes3D

A *CAxes3D* object defines the parameters necessary to draw 3D axes on the screen. Objects of this type are created by calling the *NewAxes* member function of a *CChart3D* object. Each time the *NewAxes* member function is called, a new *CAxes3D* object is created and added to the linked list maintained by the parent *CChart3D* object. Axes objects do not define the scale used in a graph, the *CTransform3D* object does. The *CAxes3D* object defines the limits of the axes in the scaling units of the associated *CTransform3D* object. *CAxes3D* objects have many properties associated with the lines and tick marks used to draw the axes, and the numeric or string labels used to label the axes tick marks.

CChartObj3D

Bar graphs, area fill graphs, surface charts, i.e. all of the data display routines for a 3D graph, are represented by an object of type *CChartObj3D*. Objects of this type are created by calling the *NewChartObj* member function of a *CChart3D* object. Each time the *NewChartObj* member function is called, a new *CChartObj3D* object is created and added to the linked list maintained by the parent *CChart3D* object. A *CChartObj3D* object contains a copy of all of the data (XY data or XYZ data) necessary to draw the graph. The *CChartObj3D* object also maintains a copy of all other attribute information associated with the chart object. These attributes include color information, justification, symbol size and independent axis direction.

A *CChartObj3D* is rendered in 3D space using the properties stored in the associated *CTransform3D* object. *CChartObj3D* objects in a *CChart3D* object can all reference a single *CTransform3D* object, or they can reference multiple *CTransform3D* objects.



Different chart types can be combined in the same graph.

CTextObj3D

Arbitrary text can be placed in a CChart3D object. Objects of this type are created by calling the NewTextObj member function of a CChart3D object. Each time the NewTextObj member function is called, a new CTextObj3D object is created and added to the linked list maintained by the parent CChart3D object. CTextObj3D objects are positioned in 3D space using the coordinate system of the associated CTransform3D object. CTextObj3D attributes include size, color, location, and viewing plane.

CDIBObj3D

Windows device independent bitmaps (DIBs) can be loaded and placed in 3D graphs. Objects of this type are created by calling the NewDIBObj member function of a CChart3D object. Each time the NewDIBObj member function is called, a new CDIBObj3D object is created and added to the linked list maintained by the parent CChart3D object. CDIBObj3D objects are positioned in 3D space using the coordinate system of the associated CTransform3D object. A CDIBObj3D can be scaled to any size within the current chart. The bitmap can also be repeated in 2 dimensions, creating a "tiled" or textured appearance.

Text Output for 3D Charts

There are many ways to output text in a graph. The most common use of text in 3D graphs are the numerics and strings for labeling axis tick marks. These axis labels are created using member functions of CAxes3D objects.

Legends are another form of text output for a 3D graph. Legends are created using the NewLegend3D member function of the CChartView3D class. The resulting CLegend3D object can hold an unlimited number of text strings and symbols for use in identifying the elements of a graph.

Every CChartView3D window and CChart3D window can have two headers and one footer. Both of these objects include a public pointer to a CHeader3D object which can be used to customize the headers or footers for a given graph.

The CChart3D class also includes a linked list of CTextObj3D objects. The NewTextObj member function of CChart3D can be used to create an unlimited number of text objects for a graph. These text objects are positioned in the current graph using the transform coordinate system so they can be used to label objects within the graph.

Plotting Data in a 3D Chart

3D charts are divided into 3 major categories; 2 ½ D, 3D and poly surface charts. A 2 ½ D chart is a pseudo 3D chart where a 2D chart is viewed using 3D bars or 3D area fills in a 3D viewing system. The functions which plot 2 ½ D charts have names ending with the characters 25D. A true 3D chart deals with either an array of 3D points (point3Dtype), or a poly surface (CPolysurface) type. The 3D charting functions which use an array of 3D points end in 3D. The poly surface charting and manipulation routines include polysurface in their name.

2 ½ D Charting Functions

A 2 ½ D chart handles 2D data represented by an array of independent variables (x values in more traditional 2D graphs) and an array of dependent variables (y values in 2D graphs). Multiple 2 ½ D plots can be combined into a single chart creating the "stacked" or "layered" look of presentation quality charts created with high end charting application packages. There are simple 2 ½ D chart types, group chart types and compound chart types. Plotting functions in the 2 ½ D category include:

Simple 2 ½ D Chart Types

- AreaFillChart25D** Plot a 2D set of data as a 3D area fill graph.
- AreaRangeChart25D** Plot a 2D set of data as a 3D area range graph.
- BarChart25D** Plot a 2D set of data as a 3D bar graph.
- LineChart25D** Plot a 2D set of data as a 3D line graph.
- PlaneChart25D** Plot a 2D set of data as a 3D plane chart.
- RibbonChart25D** Plots a 2D set of data as a 3D ribbon chart.
- FloatingBarChart25D** Plots a 2D set of data as a 3D floating bar chart.

Group and Stack 2 ½ D Chart Types

- GroupAreaFillChart25D** Plot a 2D set of data (1 x-array and multiple y-arrays) as a stacked 3D area fill graph.
- GroupBarChart25D** Plot a 2D set of data (1 x-array and multiple y-arrays) as either a side by side or stacked 3D bar graph.
- GroupRibbonChart25D** Plot a 2D set of data (1 x-array and multiple y-arrays) as a stacked 3D ribbon chart.

Compound Chart Objects

- BoxAndWhiskerChart25D** Plot a box and whisker 3D graph.
- OHLChart25D** Plot Open-High-Low-Close data as a 3D graph.

3D Charting Functions

The 3D charting functions plot data represented by an array of 3D values (**point3Dtype**), where each element contains x, y, and z values. An array of 3D points can be plotted as a contiguous line using the **LineChart3D** function or as a scatter plot using the **ScatterChart3D** function. A variety of 3D symbols can be used with the **ScatterChart3D** function. Plotting functions in the 3D category include:

| | |
|------------------------------|--|
| BarChart3D | Plot a set of 3D points as bars with arbitrary heights in a 3D graph. |
| LineChart3D | Plot a set of 3D points as a contiguous line in a 3D graph. |
| PlaneChart3D | Plot a set of 3D points as planes in a 3D graph. |
| ProjectionLineChart3D | Plot a set of 3D points, projecting the resulting line as a 2D graph onto the X, Y or Z plane. |
| ScatterChart3D | Plot a set of 3D points as scatter plot symbols in a 3D graph. |
| ScatterText3D | Plot a set of 3D points as user specified strings in a 3D graph. |

3D Poly Surface Creation and Plotting

A poly surface is a structure (**CPolysurface**) defining a graphical object as a group of 3D polygons. The poly surface structure contains all of the raw X, Y, Z data points, and the polygon vertex lists necessary to draw a complex 3D object. The structure is dynamic and uses only as much memory as the object requires. A poly surface can contain as few as one, or an unlimited number of polygons (up to the amount of memory in the computer). A poly surface can contain polygons which have three or four vertices. Each polygon in a poly surface must be planar, regardless of the number of vertices. If you have a collection of 3D points but do not have a polygon vertex list, a surface cannot be drawn. You must define polygons for the 3D points which places the polygon surfaces in proper relation to one another. Three methods are available to create the polygon vertex list.

The first is to define every polygon manually using the function **AddPolysurfacePolygon**. This is useful only if

there are a very small number of polygons or if you can create a routine to call the function automatically with the proper parameters.

The second method involves the use of the function **ConvertRG2PS** (short for Convert Regular Grid To Poly Surface). This function takes as input a collection of 3D points spaced on a regular grid in the XY plane, and creates a poly surface, complete with polygon vertex lists.

The third method involves the use of the **Delaunay Triangulation** algorithm. Delaunay triangulation can automatically create a triangular mesh of interconnecting polygons based on an arbitrary collection of data points. This is critical for applications in cartography, finite element analysis, and contour plotting because these applications typically do not use evenly spaced points. The member function **DelaunaySurface** takes a **CPolysurface** object which has a complete surface point list and calculates the polygon vertex list which completes the poly surface. In order to use any of the contour plotting functions contained in this software, the surface polygon list making up the contour plot must also include an adjacent polygon list. The adjacent polygon list is created as a by-product of the Delaunay Triangles routine. There are many constructors for creating **CPolysurface** objects using different types of data.

Creating a CPolysurface Object

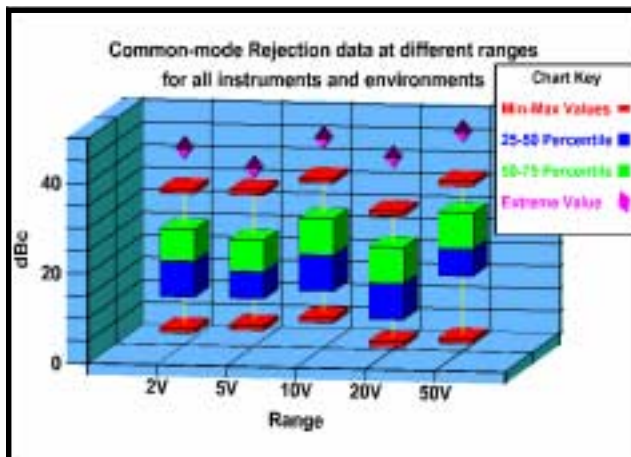
The **CPolysurface** constructor initializes a poly surface structure, dynamically allocating enough memory to hold all of the poly surface vertex points, poly surface vertex lists, and polygon colors. The functions **SetPolysurfacePoints** and **SetOnePolysurfacePoint** write 3D point entries to the poly surface point list. The function **AddPolysurfacePolygon** adds the vertex list for one polygon to the poly surface.

GetPolysurfacePolygon retrieves the 3D points associated with one poly surface polygon. The function **SetPolysurfaceColors** will set the fill color and border color for each polygon in the poly surface. If you have a set of 3D points, but not the polygon vertex list needed to complete a poly surface object, use the function **DelaunaySurface** to automatically create the polygon vertex list. The Delaunay triangulation algorithm does not require 3D points positioned on a regularly spaced grid. It is an efficient algorithm that can triangulate an arbitrary collection of points in a plane, used with contour plotting and 3D surface plotting routines in this software package.

Plotting a Poly Surface Chart

Once the definition of a poly surface object is complete, it can be plotted in a 3D chart using one of the poly surface plotting functions in the **CChart3D** class. These functions are:

| | |
|-----------------------------|---|
| ContourPolysurface3D | Plot a 3D poly surface as a 2D contour line graph projected onto one of the X, Y or Z planes. |
| PolysurfaceChart2D | Plot a 3D poly surface as a 2D projection on the X, Y or Z plane. |



A 3D Box and Whisker chart combined with a legend window.

PolysurfaceChart3D Plot a 3D poly surface as a 3D surface in a 3D graph.

The poly surface does not have to be one color since colors are defined for every polygon in the poly surface independently. A popular option in 3D surface plotting is to plot the projection of the 3D poly surface onto a plane. This can be done with the **PolysurfaceChart2D** function. Another popular option associated with poly surfaces is to plot the contours of the poly surface in a standard 2D contour plot. The 3D charting function **ContourPolysurface3D** projects the Z value contour lines of a poly surface onto the XY plane of the chart.

Independent Variable Mapping for Poly Surfaces

Independent variable mapping allows you to use color in a poly surface chart to denote a fourth dimension or variable in a chart, such as the effects pressure or temperature. Every 3D point in a **CPolysurface** object is of type **point3Dtype**, which has an element *v*, that can be used to hold the value of an independent variable. The **CChart3D** member function **SetTextureColorArray** sets the minimum and maximum values and the range of colors used for independent variable mapping. Call **SetTextureMode** to turn independent variable mapping on or off.

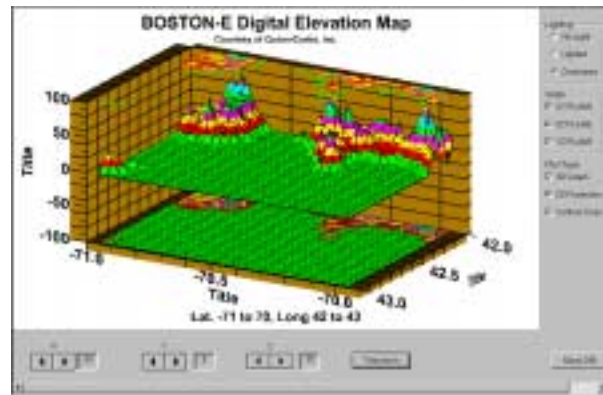
Demo Programs

Numerous example programs are provided with these tools, both on disk and in the written documentation. They illustrate usage of various classes and functions and should serve as starting points for your own applications.

Diagnostics

The class libraries can be used in both Debug and Release mode versions. Naturally, you should develop your application in debug mode first and switch to release version only when all the work is finished and debugging is completed.

Various types of errors can occur when functions of these tools are called. For example, the dynamic memory operator **new** can fail, or improper parameters may be passed to one of the member functions. In debug mode an **ASSERT** statement will typically stop the program when an error is detected. In release mode an error will be “thrown” from the point where the error is detected. You can write a “catch” routine in your main application program which takes a specific action or posts a specific message based on the location and type of error.



The 3D rendering of Boston harbor based on a Digital Elevation file.

A Summary of 3D++ Class Member Functions

Class CChartView3D

Chart3D and Legend Creation Routines

NewChart3D - Create a new **CChart3D** object and add it to the linked list of other **CChart3D** objects.

NewLegend3D - Create a new **CLegend3D** object and add it to the linked list of other **CLegend3D** objects.

View Drawing Routines

DrawViewPrinter - Draw the current **CChartView3D** object and output the result to a printer.

DrawViewScreen - Draw the current **CChartView3D** object and output the result to the screen.

Printing Routines

OnPrint - Call this function inside the **CView** class member function **OnPrint**.

SetNormPrintRect - Set the output rectangle for the printer.

SetPrintScaleMode - Set the printer scaling mode: best fit, stretch to page or scale.

View Attribute Routines

SetViewBkColor - Set the view background color.

SetViewBkMode - Set the view background mode: opaque or transparent.

SetFont - Set the TrueType font for the view.

Member functions are also available for retrieving all **CChartView3D** attributes.

Class CChart3D

Transform / Axes Creation Functions

AutoAxes - Automatically create a 3D axes object based on the current transform.

AutoTransform25D - Analyze one set of 2 ½ D data and create a transform which can be used to plot the data.

AutoTransform3D - Analyze one set of 3D data and create a transform which can be used to plot the data.

AutoTransformGroup25D - Analyze one set of 2 ½ D group data and create a transform which can be used to plot the data.

AutoTransformN25D - Analyze multiple sets of 2 ½ D data and create a transform which can be used to plot the data.

AutoTransformPolysurface - Analyze a poly surface and create a transform which can be used to plot the data.

CalcAxisTicMarks - Calculate intercepts and tick spacing for a given set of minimum and maximum values.

MinMaxValues25D - Find the minimum and maximum values of data.

NewTransform - Create a new **CTransform3D** object and add it to the linked list of other **CTransform3D** objects.

RoundAxis - Round minimum and maximum values to reasonable axis values.

Chart Object Creation Functions

NewAuxObj - Create a new **CAuxObj3D** object.

NewAxes - Create a new **CAxes3D** object.

NewChartObj - Create a new **CChartObj3D** object.

NewChartTextObj - Create a new **CTextObj3D** object.

NewDIBObj - Create a new **CDIBObj3D** object.

NewTransform - Create a new **CTransform3D** object.

Misc. Chart Functions

EnableChart - Turn the display of a chart on or off.

SetChartPosition - Set the current position of a chart in the view.

SetColorMode - Set the current color mode to no lighting, lighting or textures.

SetFontSize - Set the relative font size for a window.

SetLightPosition - Set the light position.

SetTextureColorArray - Set the colors in the texture color array.

SetTextureMode - Set the texture color mode.

Member functions are also available for retrieving all **CChart3D** attributes.

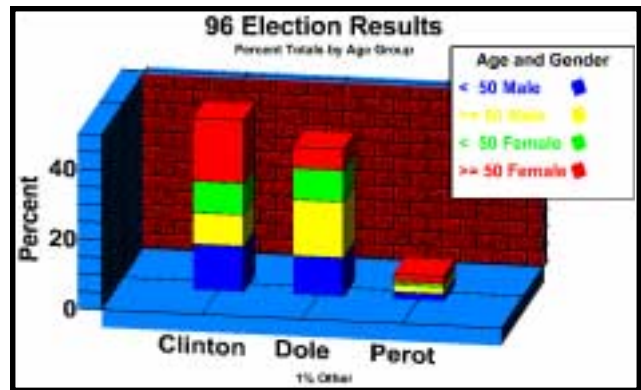
Class CAxes3D

Axis Routines

EnableAxis - Enable/Disables an axis.
minor tick marks.

SetAxisAttrib - Sets the axis attributes (color, line style, line width).

SetAxisCenterPoint - Sets the center point of the axis.
Drawing of tick marks start at the axis center point.



A red brick bitmap texture has been placed in front of the back wall.

SetAxisColor - Sets the axis color.

SetAxisIntercept - Sets the two intercepts necessary to position the axis with respect to the current transform.

SetAxisIntercept1 - Sets the first intercept of the axis with the other two axes.

SetAxisIntercept2 - Sets the second intercept of the axis with the other two axes.

SetAxesLimits - Sets the min and max limits for all three axes.

SetAxisLimits - Sets the axis min and max limits.

SetAxisMajorTicLength - Sets the axis major tick mark length.

SetAxisMax - Sets the axis maximum limit.

SetAxisMin - Sets the axis minimum limit.

SetAxisMinorTicLength - Sets the axis minor tick mark length.

SetAxisNthTicMajor - Sets the skip factor for major axis tick marks.

SetAxisStyle - Sets the axis line style (dotted, dashed, etc.).

SetAxisThickness - Sets the axis line thickness.

SetAxisTicColor - Sets the axis tick mark color.

SetAxisTicDir - Sets the axis tick direction.

SetAxisTics - Sets the center point, minor tick spacing, skip factor for major tick marks, minor tick mark length, major tick mark length, and the tick direction for an axis.

SetAxisTicSpace - Sets the space between adjacent minor tick marks for an axis.

Plane and Wall Routines

EnablePlane - Enable/Disables a plane.

EnableWall - Enable/Disables a wall or a plane.

SetOutlineWidth - Sets the outline line width.

SetOutlineMode - Sets the wall/plane outline mode on or off.

SetOutlineColor - Sets the color of plane and wall outlines.

SetPlaneColor - Sets the plane color.

SetWallOffset - Sets the offset used when placing planes within the current transform. Units used are normalized and represent a percent of the full scale value for the axis normal to the plane.

SetWallPlane - Turns on a wall and sets the wall fill and outline colors.

SetWallThickness - Sets the wall thickness. Units used are normalized and represent a percent of the full scale value for the axis normal to the plane.

Grids

EnableGrid - Enable/Disables a grid.

SetPlaneGrid - Sets the spacing, color, line style, line thickness and wall flag for a grid.

SetPlaneGridAttrib - Sets the color, line style, line thickness for a grid.

SetPlaneGridColor - Sets the grid color.

SetPlaneGridNthTic1 - Sets skip factor #1 for placing grid lines on the plane.

SetPlaneGridNthTic2 - Sets skip factor #2 for placing grid lines on the plane.

SetPlaneGridStyle - Sets the grid line style.

SetPlaneGridThickness - Sets the line thickness used in drawing the grid.

SetPlaneGridWallFlag - Sets the wall flag for a grid. The wall flag determines if the grid wraps around the associated wall.

Axis Label Routines

EnableAxisLabelsStrings - Enable/Disables labeling the axis with strings instead of numeric values. coefficient.

SetAxisLabels - Sets the axis labels plane, rotation, direction, number of decimals, end point label mode, color and optional tick strings.

SetAxisLabelsColor - Sets the axis label color.

SetAxisLabelsDecimals - Sets the precision of the numeric labels.

SetAxisLabelsDir - Sets the axis label drawing direction.

SetAxisLabelsEnds - Sets the axis label end point label mode.

SetAxisLabelsPlane - Sets the axis label plane.

SetAxisLabelsRotation - Sets the rotation of the axis label within the axis label plane.

SetAxisLabelsSize - Sets the axis label relative size coefficient.

SetAxisLabelsStrings - Sets the axis label strings which can be used in place of the default numeric strings.

SetAxisLabelsTicOffset - Sets the offset of the axis labels with respect to tick marks.

SetAxisLabelsXJust - Sets the X justification of the axis label strings.

SetAxisLabelsYJust - Sets the Y justification of the axis label strings.

SetAxisTitleColor - Sets the axis title color.

SetAxisTitlePlane - Sets the axis title plane.

SetAxisTitleSize - Sets the axis title relative size coefficient.

SetAxisTitleString - Sets the axis title string.

Texture Routines

EnableAxisTexture - Enable/Disables axis texturing.

EnablePlaneTexture - Enable/Disables wall/plane texturing.

Member functions are also available for retrieving all **CAxes3D** attributes.

Class CChartObj3D

2 1/2 D Chart Objects

AreaFillChart25D - Plot a 2D set of data as a 3D area fill graph.

AreaRangeChart25D - Plot a 2D set of data as a 3D area range graph.

BarChart25D - Plot a 2D set of data as a 3D bar graph.

BoxAndWhiskerChart25D - Plot a box and whisker 3D graph.

FloatingBarChart25D - Plot a 2D set of data (1 x-array and 2 y-arrays) as a 3D floating bar graph.

GroupAreaFillChart25D - Plot a 2D set of data (1 x-array and multiple y-arrays) as a stacked 3D area fill graph.

GroupBarChart25D - Plot a 2D set of data (1 x-array and multiple y-arrays) as either a side by side or stacked 3D bar graph.

GroupRibbonChart25D - Plot a 2D set of data (1 x-array and multiple y-arrays) as a stacked 3D ribbon chart.

LineChart25D - Plot a 2D set of data as a 3D line graph.

OHLCCChart25D - Plot Open-High-Low-Close data as a 3D graph.

PlaneChart25D - Plot a 2D set of data as a 3D plane chart.

RibbonChart25D - Plot a 2D set of data as a 3D ribbon chart.

3D Chart Objects

BarChart3D - Plot a set of 3D points as bars with arbitrary heights in a 3D graph.

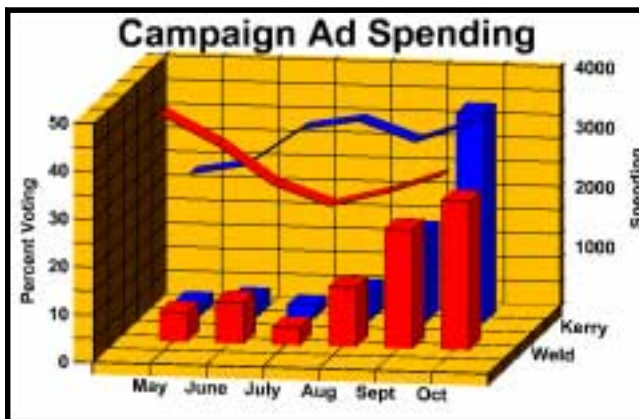
LineChart3D - Plot a set of 3D points as a contiguous line in a 3D graph.

PlaneChart3D - Plot a set of 3D points as planes in a 3D graph.

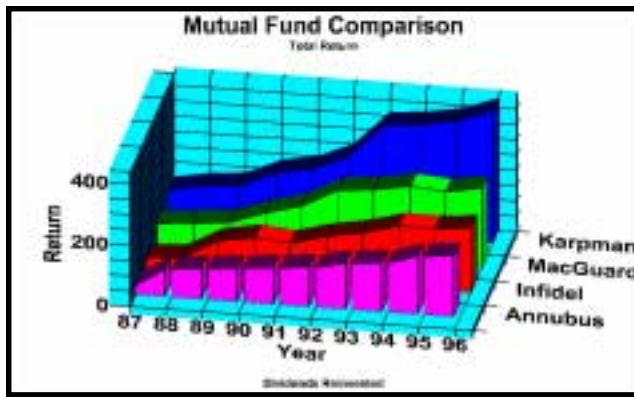
ProjectionLineChart3D - Plot a set of 3D points, projecting the resulting line as a 2D graph onto the X, Y or Z plane.

ScatterChart3D - Plot a set of 3D points as scatter plot symbols in a 3D graph.

ScatterText3D - Plot a set of 3D points as user specified strings in a 3D graph.



This graph has two Z-Axes, each with a different scaling.



Axis labels use 3D characters. Axes can be labeled with numerics or string literals.

3D Polysurface Objects

ContourPolysurface3D - Plot a 3D poly surface as a 2D contour line graph projected onto one of the X, Y or Z planes.

PolysurfaceChart2D - Plot a 3D poly surface as a 2D projection on the X, Y or Z plane.

PolysurfaceChart3D - Plot a 3D poly surface as a 3D surface in a 3D graph.

Chart Object Attribute Functions

EnableChartObj - Turn on/off the chart object.

SetBaseValue - Set the base value for a 3D bar graph data point.

SetChartType - Set the chart object graph type.

SetClipping - Set the state of the chart object clipping flag.

SetComboChartFlags - Set the combo chart flags.

SetConstJustify - Set the value of the constant justify parameter for a 2 ½ D chart object.

SetConstValue - Set the value of the constant parameter for a 2 ½ D chart object.

SetConstWidth - Set the value of the constant width for a 2 ½ D graph.

SetContourColor - Set the color of a specific contour level in a contour chart object.

SetDepJustify - Set the value of the dependent justify parameter for a 2 ½ D chart object.

SetDepWidth - Set the value of the dependent width for a 2 ½ D graph.

SetFillColor - Set the fill color for an area fill chart object.

SetFillGap - Set the fill gap for an area fill chart object.

SetGroupColor - Set the color for a specific group in a group chart object.

SetGroupType - Set the group type for a group chart object.

SetIndAxis - Set the independent axis for a 2 ½ D chart object.

SetIndJustify - Set the value of the independent justify parameter for a 2 ½ D chart object.

SetIndWidth - Set the value of the independent width for a 2 ½ D graph.

SetLineStyle - Set the line style for a line chart object.

SetLineWidth - Set the line width for a line chart object.

SetOutlineMode - Set the state of the chart object outline mode.

SetOutlineWidth - Set the thickness of the lines used for outlines.

SetPlaneThickness - Set the thickness of planes in the plane chart objects.

SetProjectionPlane - Set the projection plane for a projection chart object.

SetRelSize - Set the relative size for scatter plot symbols in a scatter plot chart object.

SetSegmentColor - Set the segment color for a chart object.

SetSegmentColorMode - Set the state of the segment color mode.

SetSymbol - Set the scatter plot symbol in a scatter plot chart object.

SetTextureEnable - Set the state of the chart object texture mode.

Chart Object Data Functions

Set3DPoint - Set the 3D data point at a specific index for a 3D chart object.

SetBarHeight - Set the bar height at a specific index for a 3D bar graph data point.

SetContourValue - Set the value of a specific contour level in a contour chart object.

SetDepData - Set the value at a specific index for the dependent data in a 2 ½ D chart object.

SetDepData2 - Set the value at a specific index for the second set of dependent data in a 2 ½ D chart object.

SetGroupData - Set the value at a specific index for group data in a 2 ½ D chart object.

SetIndData - Set the value at a specific index for the independent data in a 2 ½ D chart object.

SetNumContours - Set the number of contour levels in a contour chart object.

SetNumDataPnts - Set the number of data points in a chart object.

SetNumGroups - Set the number of groups in a group chart object.

SetPolysurfacePoint - Set a single poly surface point for a poly surface chart object.

Member functions are also available for retrieving all **CChartObj3D** attributes.

Class CTransform3D

SetAxisScaleMode - Set the axis scaling mode to linear or logarithmic.

SetChartRect - Scale the transform within the **CChart3D** view.

SetOffset - Position the transform within the **CChart3D** view.

SetRotate - Set the transform rotational parameters.

SetScale - Set the minimum and maximum X, Y and Z values for the transform physical coordinate system.

SetScaleMin - Set the minimum X, Y and Z values for the transform physical coordinate system.

SetScaleMax - Set the maximum X, Y and Z values for the transform physical coordinate system.

NormalizePoint - Use the transform to convert from physical units to normalized units.

UnNormalizePoint - Convert a point in normalized coordinates to physical coordinates.

Member functions are also available for retrieving all **CTransform3D** attributes.

Class CPolysurface

AddPolysurfacePolygon - Add a polygon to the **CPolysurface** polygon list.

CopyXYorZtoV - Copy the x, y or z coordinate of a poly surface point to the v coordinate. This a useful function if you want to texture a poly surface which is projected onto a plane.

CreateRGAdjacentPolygonList - Create an adjacent polygon list for a poly surface object based on an evenly spaced grid.

DelaunaySurface - Use the Delaunay algorithm to create the polygon edge list based on the 3D point list in a **CPolysurface** object.

ResetPolysurfacePolygons - Reset the polygon edge list to empty. The 3D point list is unaffected.

Save - Save a poly surface as a file.

SetOnePolysurfacePoint - Set a specific point in the poly surface point list.

SetPolysurfacePoints - Set a group of points in the poly surface point list.

SetPolysurfaceColors - Set the colors for a poly surface polygon.

Member functions are also available for retrieving all **CPolysurface** attributes.

Class CLegend3D

ContourLegend - Create a legend for a contour plot where the contour range is displayed as a color.

ContourLegend2 - Create a legend for a contour plot where contour values are mapped as lines in a 3D graph.

EnableLegend - Turn the legend on/off.

NewLegendObj - Create a new **CLegendObj3D** object.

SetColor - Set the color for a legend item object.

SetFooter - Set the footer string.

SetFontSize - Set the legend font size.

SetLegendBkColor - Set the legend background color.

SetLegendBorderColor - Set the legend outline color.

SetLegendColumns - Set the number of columns in a legend.

SetLegendRows - Set the number of legend rows.

SetLegendPosition - Set the position of the legend in the **CChartView3D** window.

SetBorderThickness - Set the thickness of the legend outline.

SetSymbol - Set the symbol for a specific legend item.

SetSymbolColor - Set the color of a legend symbol.



3D charts can be placed in **CFormView** windows.

SetSymbolScale - Set the scaling for a legend symbol.

SetText - Set the string for a legend item.

SetTextColor - Set the color of the text for a legend item.

SetTitle1 - Set the first title string for a legend.

SetTitle2 - Set the second title string.

Member functions are also available for retrieving all **CLegend3D** attributes.

Class CTextObj3D

EnableTextObj - Turn on/off the display of a text object.

OutText - Draw the text object using physical coordinates.

OutTextNorm - Draw the text object using normalized coordinates.

SetCharSize - Set the character size of the text object.

SetColor - Set the color of the text object.

SetJustify - Set the x and y justification parameters.

SetLocation - Set the text object position using physical coordinates.

SetNormLocation - Set the text object location using normalized coordinates.

SetRotation - Set the text object rotation within the viewing plane.

SetString - Set the string displayed by the text object.

SetTextPlane - Set the viewing plane of the text object.

SetXJustify - Set the text object x justification.

SetYJustify - Set the text object y justification.

TextWidth - Get the width of the text object in normalized coordinates.

TextHeight - Get the height of the text object in normalized coordinates.

Member functions are also available for retrieving all **CTextObj3D** attributes.

Class CAuxObj3D

SetColor - Set the aux object color.
SetAuxObjType - Set the aux object type.
SetLineThickness - Set the aux object line thickness.
SetLocation - Set the position of the aux object in physical coordinates.
SetNormLocation - Set the position of the aux object in normalized coordinates.
SetRotation - Set the rotation of the aux object.
SetScaleFactor - Set the aux object relative scale factor.
SetSizeParameters - Set the aux object size parameter.
SetSolidType - Set the draw mode to solid or wireframe.

Member functions are also available for retrieving all **CAuxObj3D** attributes.

Class CDIBObj3D

SetDIBFilename - Set the file name of a DIB bitmap.
SetNormPlanePoints - Specify three points in normalized coordinates which form the plane of the DIB object.
SetPlanePoints - Specify three points in physical coordinates which form the plane of the DIB object.
SetRepeatFactors - Set the repeat factors in the x and y direction for the DIB object.

Member functions are also available for retrieving all **CDIBObj3D** attributes.

Example Programs

The source code listings for several of our 3D++ example programs are available from our FTP site at <ftp.webcom.com> in `/pub/quinn/demofiles`, or from our BBS at (617)449-4783. The file names are EX1.ZIP, SURF.ZIP and MUTF.ZIP. These files are already installed on your computer in the \EXAMPLES subdirectory if you installed our 3D++ class library demo program 3D++DEMO.



The **ScatterText3D** chart object substitutes text strings for scatter plot symbols.

Technical Support

Quinn-Curtis, Inc. provides free technical support only to individual registered users (not companies) for a period of 12 months from the date of purchase unless appropriate updates to the software have been purchased separately. Technical support beyond one year costs an additional \$200/year. This cost includes an automatic upgrade to the latest version of the software.

