

# Graphics Server demonstration

## Contents

What is Graphics Server?

How to use this demonstration

The client/server model

Window management

Co-ordinate system

Drawing functions

Text functions

Graphing functions

AutoGraph functions

Statistics functions

How data is passed

DLL interface

DDE interface

Mouse interaction

Image exchange

Availability

## **What is Graphics Server?**

### **Overview**

Graphics Server is a graphical subsystem for Windows that provides graphing and charting services for client applications.

Those client applications may be programs written in traditional programming languages such as C and Pascal, applications generated by a 4GL such as SQLWindows, or even proprietary applications such as Microsoft Word for Windows and Excel.

Graphics Server provides services to any client software with DLL or DDE capability.

Using Graphics Server, graphs and charts can be bound seamlessly into an application.

The Graphics Server run-time files can be distributed licence-free!

### **Client applications**

The Graphics Server SDK provides standard interfaces to:

- C and C++
- Borland Turbo Pascal for Windows
- Gupta SQLWindows
- SPC Superbase
- Microsoft Visual Basic
- Actor
- Powersoft PowerBuilder

Using the SDK you can create links to most client applications.

## **How to use this demonstration**

This demonstration is a Microsoft C 7.0 program developed using the Microsoft Foundation Classes and the Graphics Server SDK.

It illustrates some of the features of Graphics Server, which can enliven and enhance applications.

### **Graph menu**

Pick a graph type from the Graph menu.

The graph can be presented in various styles which you select from the Style menu.

### **Style menu**

Graphs can be customised in many ways, using colours, patterns, shadows and other drawing tools to enliven appearance.

Choose a style option to see how changing the Graphics Server colour palette can change the appearance of a graph.

**The client/server model**

Graphics Server is capable of supporting many clients simultaneously.

Each client may open and draw into many Graphics Server windows.

Graphics Server is not a server in the sense of operating across a LAN. Rather it is a server in providing a service to many clients on a workstation.

## **Window management**

One of the requirements of writing Windows applications is to repaint windows after they have been covered or re-sized.

Graphics Server removes the burden of this by maintaining its own windows independently of the client application.

The client application need only be concerned with dictating the content of the window in terms of text, objects and graphs.

Should the client application wish to manage images itself, these can be passed to it by Graphics Server in the form of metafiles.

The client application is then at liberty to play them into its own windows or printer device contexts.

## **Co-ordinate system**

Graphics Server simplifies graphical programming by presenting a consistent, device-independent logical co-ordinate system.

Within this co-ordinate system, you can place the graphical objects that compose a complete image using positions which remain constant for different screen adapters, plotters or printers.

The units of this co-ordinate system can be independently defined.

Furthermore within each window, sub-windows or views can be defined to allow easy manipulation of regions of the screen.

## **Drawing functions**

Graphics Server provides a comprehensive range of drawing tools ranging from simple functions to draw a line or arc, to complex functions for drawing filled polygons and 3-D boxes.

These functions are all expressed in the logical co-ordinate system.

Attributes such as colour, fill pattern and raster-operation (XOR etc.) can also be set, and there are functions to work with polar co-ordinates.

## **Text functions**

Graphics Server provides a wide range of text functions, handling raster and vector fonts, and allowing selection of attributes such as colour, font face, bold, italic etc.

In addition to the functions for output of single strings, there are additional functions for labels and legends that work with arrays.



## **Graphing functions**

This demonstration shows the principal types of graphs produced by Graphics Server.

All the examples in this demonstration use the AutoGraph feature to draw their graph.

In both the custom and AutoGraph functions data is passed in arrays of integer or floating point numbers. Legend and label text is passed in string arrays.

## **AutoGraph functions**

AutoGraph produces a complete presentation of a graph inside a view using just numeric and textual graph data supplied by your application. It automatically decides such things as what axis range and scale is required to best represent your data, what font sizes are required to best draw your graph titles and labels, and so on. Your application can forget about drawing co-ordinates and concentrate on providing the data you want presented in graphical form.

The main objective of AutoGraph is to be easy to use. Sometimes this means that AutoGraph is unable to cope with very specialised graphing requirements. In such cases you can bypass AutoGraph and call upon the custom graphing functions of Graphics Server. These functions are the same ones used by AutoGraph but, by calling them yourself, you take over the precise control of the layout and style of your graph.

Using information returned by AutoGraph about a graph it has drawn on your behalf you can use the custom graphing functions to superimpose another graph. This is the technique used by two of the examples in this demonstration; Line with Area and Line with Bar.

## **Statistics functions**

Once a graph has been drawn, Graphics Server can be requested to add various statistical lines calculated from the data.

These include minimum, maximum, mean, standard-deviation and linear regression best-fit.

## **How data is passed**

The data for graphing can come from a spreadsheet, database or by numerical calculation.

There is no restriction on how the data is derived. Graphics Server simply deals with arrays of values.

These arrays may either be passed one value at a time in a series of calls, or in one operation to pass a complete array.

## **DLL interface**

Graphics Server presents a DLL interface to client programs. The DLL interface to Graphics Server follows the broadly accepted standard for such interfaces under Windows and is capable of being called from a wide range of applications. Examples of applications able to call the DLL interface include those written in C, SQLWindows and Superbase.

The Graphics Server DLL interface uses the same FAR PASCAL calling convention employed by Windows itself. In addition to setting up the correct calling convention, a DLL client must also deal correctly with:

- \* Passing integers, 8-byte doubles and far pointers to null-terminated character strings, together with arrays of these types, as arguments to a function.
- \* Receiving an integer or double return value from a function.

## **DDE interface**

As well as the procedure call interface to Graphics Server, provided by the DLL, there is a message based interface provided through the mechanism of Dynamic Data Exchange (DDE). The DDE interface is ideal for clients without a DLL calling capability.

Graphics Server responds to a DDE Initiate request from a client application by opening a DDE dialogue through which the client can call the normal API functions of the server.

Once a DDE dialogue is open the client Executes a Graphics Server API function by passing a text form of the standard function call in a DDE message. The client can also Request the result of calling a function, again in text form, in another DDE message.

When the client is finished using Graphics Server it Terminates the DDE dialogue. This automatically closes the graphics windows opened on its behalf.

A client application only needs a single DDE dialogue with Graphics Server to open multiple graphics windows and views. Graphics Server will hold simultaneous dialogues with as many DDE clients as required.

## **Mouse interaction**

Graphics Server allows you to nominate arbitrary regions in a window as hot-regions.

When a mouse event occurs in such a region a message is returned to the client application, defining the event and the region.

In the most sophisticated extreme, your application can make one function call to request that all the points in any subsequent graphs are automatically made into hot-regions. This is called the hot-graph facility. Once a hot-graph has been drawn your application will receive events giving point and data set information whenever the user clicks on a point in the graph.

By this means a graph can be made to function as a sophisticated control in an application window.

## **Image exchange**

In the world of Windows, exchange of data between applications is of paramount importance.

Graphics images are no less important as data objects than numbers or text, and Graphics Server provides means of exporting metafiles and device-independent bitmaps of its images to other applications, firstly through the clipboard and secondly through permanent files.

Images in the clipboard can be pasted directly into other documents.

Images in a file can be opened and read in another application.

The opening screen of this demonstration is a device-independent bitmap created and exported by Graphics Server.



## Area graph

```
void CGSMainWnd::GraphArea2D()
{
    char    FAR* strTitleg = "Total sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "Units sold";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA, AGAREAYGRID, 0 );

    AGClose();
}
```

## Absolute area graph

```
void CGSMainWnd::GraphArea2DAbsolute()
{
    char    FAR* strTitleg = "Regional sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "Units sold";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA, AGAREAABS | AGAREAYGRID, 0 );

    AGClose();
}
```

## Percentile area graph

```
void CGSMainWnd::GraphArea2DPercent()
{
    char    FAR* strTitleg = "Proportion of sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "%";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA, AGAREAPC | AGAREAYGRID, 0 );

    AGClose();
}
```

### 3D area graph

```
void CGSMainWnd::GraphArea3D()
{
    char    FAR* strTitleg = "Total operating costs";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrDeptNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA3D, AGAREAYGRID, 0 );

    AGClose();
}
```

### 3D absolute area graph

```
void CGSMainWnd::GraphArea3DAbsolute()
{
    char    FAR* strTitleg = "Regional manufacturing costs";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA3D, AGAREAABS | AGAREAYGRID, 0 );

    AGClose();
}
```

### 3D percentile area graph

```
void CGSMainWnd::GraphArea3DPercent()
{
    char    FAR* strTitleg = "Proportion of marketing costs";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "%";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA3D, AGAREAPC | AGAREAYGRID, 0 );

    AGClose();
}
```

## Bar graph

```
void CGSMainWnd::GraphBar2D()
{
    char    FAR* strTitleg = "Product support costs";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrDeptNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR2D, AGBARCLUSTER | AGBARYGRID, 0 );

    AGClose();
}
```

## Horizontal bar graph

```
void CGSMainWnd::GraphBar2DHorz()
{
    char    FAR* strTitleg = "Total administration costs";
    char    FAR* strTitlex = "$";
    char    FAR* strTitley = "1992";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR2D, AGBARSTACK | AGBARHORIZ | AGBARXGRID, 0 );

    AGClose();
}
```



## Percentile bar graph

```
void CGSMainWnd::GraphBar2DPercent()
{
    char    FAR* strTitleg = "Operating budget allocation";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "%";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrDeptNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR2D, AGBARSTACKPC | AGBARYGRID, 0 );

    AGClose();
}
```

## 3D bar graph

```
void CGSMainWnd::GraphBar3D()
{
    char    FAR* strTitleg = "Regional sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR3D, AGBARCLUSTER | AGBARYGRID, 0 );

    AGClose();
}
```

### 3D horizontal bar graph

```
void CGSMainWnd::GraphBar3DHorz()
{
    char    FAR* strTitleg = "Product sales";
    char    FAR* strTitlex = "$";
    char    FAR* strTitley = "1992";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR3D, AGBARSTACK | AGBARHORIZ | AGBARXGRID, 0 );

    AGClose();
}
```

### 3D percentile bar graph

```
void CGSMainWnd::GraphBar3DPercent()
{
    char    FAR* strTitleg = "Proportion of total sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "%";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR3D, AGBARSTACKPC, 0 );

    AGClose();
}
```

### 3D Z-clustered bar graph

```
void CGSMainWnd::GraphBar3DZ()
{
    char    FAR* strTitleg = "New product sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR3D, AGBARCLUSTZ | AGBARYGRID, 0 );

    AGClose();
}
```

## Bubble graph

```
void CGSMainWnd::GraphBubble()
{
    char    FAR* strTitleg = "Bubble graph";
    char    FAR* strTitlex = "x";
    char    FAR* strTitley = "y";

    double  fRadius [NUMMONTHS];
    double  fDist    [NUMMONTHS] [2];
    int     BubbleNum;

    for ( BubbleNum = 0; BubbleNum < NUMMONTHS; ++BubbleNum ) {
        fRadius[BubbleNum] = fmod( rand(), 500 );
        fDist[BubbleNum][0] = fmod( rand(), 8000 );
        fDist[BubbleNum][1] = fmod( rand(), 8000 );
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fRadius );
    AGDist( NUMMONTHS * 2, &fDist[0][0] );
    AGTitleG( strTitleg );
    AGTitlex( strTitlex );
    AGTitley( strTitley );
    AGXAxisStyle( AGUTICKS | AGUMAX, 5, 0, 10000, 0 );
    AGYAxisStyle( AGUTICKS | AGUMAX, 5, 0, 10000, 0 );
    SelectFonts();

    AGShow( AGBUBBLE, 0, 0 );

    AGClose();
}
```

## Curve fit

```
void CGSMainWnd::GraphCurveFit()
{
    char    FAR* strTitleg = " regional profitability curve";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    char    strTitle [60];
    double  fProfit [NUMMONTHS];
    double  fDist   [NUMMONTHS];
    int     Patt    [1];
    int     Symbol   [1];
    int     MonthNum, RegionNum;

    RegionNum = rand() % NUMREGIONS;
    lstrcpy( strTitle, g_lpstrRegionNames[RegionNum] );
    lstrcat( strTitle, strTitleg );

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fProfit[MonthNum] = g_fSales[MonthNum][RegionNum]
            - g_fCosts[MonthNum][RegionNum];

        fDist[MonthNum] = MonthNum;
    }

    Patt[0] = 2; Symbol[0] = 3 + 2 * RegionNum;

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fProfit );
    AGDist( NUMMONTHS, fDist );
    AGPatt( 1, Patt );
    AGSym( 1, Symbol );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGTitleG( strTitle );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    AGCurveStyle( CFPOLY, NUMMONTHS / 2 + 1, 50 );
    AGXAxisStyle( AGAXISBOTTOM, 0, 0, 0, 0 );
    SelectFonts();

    AGShow( AGSCATTER, AGSCATTSYMBOLS | AGSCATTTHICK | AGSCATTCURVE, 0 );

    AGClose();
}
```

## Gantt chart

```
void CGSMainWnd::GraphGantt()
{
    char    strTitleg [60], FAR* pstrTitleg = "Production schedule";
    char    strTitlex [20], FAR* pstrTitlex = "Period";
    char    strTitley [20], FAR* pstrTitley = "Item";

    OemToAnsi( pstrTitleg, strTitleg );
    OemToAnsi( pstrTitlex, strTitlex );
    OemToAnsi( pstrTitley, strTitley );

    double  fAmp [3] [4];

    static const char FAR* lpstrItemNames [3] = {
        "Key", "Plate", "Shaft"
    };

    static const char FAR* lpstrActivityNames [3] = {
        "Tooling", "Prototyping", "Pre-production"
    };

    int      RegionNum, ItemNum;

    for ( RegionNum = 0; RegionNum < NUMREGIONS; ++RegionNum ) {
        fAmp[RegionNum][0] = g_fCosts[0][RegionNum] / 1000;

        for ( ItemNum = 1; ItemNum < 4; ++ItemNum ) {
            fAmp[RegionNum][ItemNum] = fAmp[RegionNum][ItemNum - 1]
                + g_fCosts[ItemNum][RegionNum] / 1000;
        }
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGAmp( 3, 4, &fAmp[0][0] );
    AGLabels( 3, lpstrItemNames );
    AGLegend( 3, lpstrActivityNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    AGYAxisStyle( AGLABEVERY | AGTICKEVERY, 5, 5, 0, 0 );
    SelectFonts();

    AGShow( AGGANTT, AGGANTTXGRID | AGGANTTSPACE, 0 );

    m_GraphAxisLowVal = 0;
    AGClose();
}
```



## High-low-close graph

```
void CGSMainWnd::GraphHLC()
{
    char    FAR* strTitleg = "Commodity price";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "Pence";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, HLCGROUPSIZE, &g_fPrice[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGHLC, AGHLCNOBARS | AGHLCTHICK, AGMEAN | AGMINMAX | AGSD );

    AGClose();
}
```

## Line graph

```
void CGSMainWnd::GraphLine()
{
    char    FAR* strTitleg = "Sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    int     Patt    [NUMREGIONS];
    int     Symbol  [NUMREGIONS];
    int     RegionNum;

    for ( RegionNum = 0; RegionNum < NUMREGIONS; ++RegionNum ) {
        Patt[RegionNum] = 2; Symbol[RegionNum] = 3 + 2 * RegionNum;
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    AGOpen();
    AGAmp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGPatt( NUMREGIONS, Patt );
    AGSym( NUMREGIONS, Symbol );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGLINE, AGLINESYMBOLS | AGLINESOLID | AGLINETHICK, 0 );

    AGClose();
}
```

## Straight line fit

```
void CGSMainWnd::GraphLineFit()
{
    char    FAR* strTitleg = " regional profitability line";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    char    strTitle [60];
    double  fProfit [NUMMONTHS];
    double  fDist   [NUMMONTHS];
    int     Patt    [1];
    int     Symbol  [1];
    int     MonthNum, RegionNum;

    RegionNum = rand() % NUMREGIONS;
    strcpy( strTitle, g_lpstrRegionNames[RegionNum] );
    strcat( strTitle, strTitleg );

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fProfit[MonthNum] = g_fSales[MonthNum][RegionNum]
            - g_fCosts[MonthNum][RegionNum];

        fDist[MonthNum] = MonthNum;
    }

    Patt[0] = 2; Symbol[0] = 3 + 2 * RegionNum;

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fProfit );
    AGDist( NUMMONTHS, fDist );
    AGPatt( 1, Patt );
    AGSym( 1, Symbol );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGTitleG( strTitle );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    AGCurveStyle( CFPOLY, 1, 50 );
    AGXAxisStyle( AGAXISBOTTOM, 0, 0, 0, 0 );
    SelectFonts();

    AGShow( AGSCATTER, AGSCATTSYMBOLS | AGSCATTTHICK | AGSCATTCURVE, 0 );

    AGClose();
}
```

## Line overlaid on area graph

```
void CGSMainWnd::GraphLineWithArea()
{
    char FAR* strTitleg = "Commodity price variation";
    char FAR* strTitlex = "1992";
    char FAR* strTitley = "Pence";

    double fHL [NUMMONTHS] [2];
    double fC [NUMMONTHS];
    int Patt [2];
    int Symbol [2];
    int Clr [2];
    int MonthNum;

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fHL[MonthNum][0] = g_fPrice[MonthNum][HLCLOW];
        fHL[MonthNum][1] = g_fPrice[MonthNum][HLCHIGH] - g_fPrice[MonthNum][HLCLOW];
        fC[MonthNum] = g_fPrice[MonthNum][HLCCLOSE];
    }

    Patt[0] = BRNULL; Clr[0] = 0;
    Patt[1] = BRSOLID; Clr[1] = LIGHT | BLUE;

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGAmp( NUMMONTHS, 2, &fHL[0][0] );
    AGPatt( 2, Patt );
    AGClr( 2, Clr );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGAREA, 0, 0 );

    Patt[0] = 2; Symbol[0] = 5; Clr[0] = LIGHT | WHITE;

    GSDataDim( NUMMONTHS, 1 );
    GSDataAmp( NUMMONTHS, 1, fC );
    GSDataScale( AGInfo( 5 ) / AGInfo( 2 ) );
    GSDataPatt( 1, Patt );
    GSDataSym( 1, Symbol );
    GSDataClr( 1, Clr );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    GSXYGraph( AGInfo( 6 ),
               AGInfo( 7 ),
               AGInfo( 4 ) / (NUMMONTHS - 1),
               XYGLINE | XYGSYMBOL | XYGTHICK | XYGGROUPED,
               0 );

    GSAXis( AGInfo( 6 ) + AGInfo( 4 ),
            AGInfo( 7 ),
            AGInfo( 5 ),
            20,
            1,
            0,
            AXTICKTHRU | AXISY,
            LSSOLID,
            m_StyleFG );

    AGClose();
}
```

## Line overlaid on bar graph

```
void CGSMainWnd::GraphLineWithBar()
{
    char FAR* strTitleg = "Sales variation with price";
    char FAR* strTitlex = "1992";
    char FAR* strTitley = "$";

    double fPrice [NUMMONTHS];
    double fSales [NUMMONTHS];
    int Patt [1];
    int Symbol [1];
    int Clr [1];
    int MonthNum, RegionNum;

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fPrice[MonthNum] = g_fPrice[MonthNum][HLCCLCLOSE];
        fSales[MonthNum] = 0;

        for ( RegionNum = 0; RegionNum < NUMREGIONS; ++RegionNum ) {
            fSales[MonthNum] += g_fSales[MonthNum][RegionNum];
        }
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fPrice );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGBAR2D, AGBARSTACK, 0 );

    Patt[0] = 2; Symbol[0] = 7; Clr[0] = LIGHT | CYAN;

    GSDataDim( NUMMONTHS, 1 );
    GSDataAmp( NUMMONTHS, 1, fSales );
    GSDataScale( AGInfo( 5 ) / 25000 );
    GSDataPatt( 1, Patt );
    GSDataSym( 1, Symbol );
    GSDataClr( 1, Clr );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    GSXYGraph( AGInfo( 6 ) + AGInfo( 4 ) / NUMMONTHS / 2,
               AGInfo( 7 ),
               AGInfo( 4 ) / NUMMONTHS,
               XYGLINE | XYGSYMBOL | XYGTHICK | XYGGROUPED,
               0 );

    GSAxis( AGInfo( 6 ) + AGInfo( 4 ),
            AGInfo( 7 ),
            AGInfo( 5 ),
            20,
            5,
            0,
            AXTICKTHRU | AXISY,
            LSSOLID,
            m_StyleFG );

    GSLabelnY( AGInfo( 6 ) + AGInfo( 4 ) + 20,
               AGInfo( 7 ),
               AGInfo( 5 ) / 5,
               0, 0,
               0, 25 / 5,
               0, 5 + 1,
               CSUSER | CSRASTER,
               TXLEFT | TXBASELINE | TXTRANS,
```

```
        m_StyleFG );  
    AGClose();  
}
```

## Log-linear graph

```
void CGSMainWnd::GraphLogLin()
{
    char    FAR* strTitleg = "Sales";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    int     Patt    [NUMREGIONS];
    int     Symbol  [NUMREGIONS];
    int     RegionNum;

    for ( RegionNum = 0; RegionNum < NUMREGIONS; ++RegionNum ) {
        Patt[RegionNum] = 2; Symbol[RegionNum] = 3 + 2 * RegionNum;
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    AGOpen();
    AGAmp( NUMMONTHS, NUMREGIONS, &g_fSales[0][0] );
    AGPatt( NUMREGIONS, Patt );
    AGSym( NUMREGIONS, Symbol );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGLOGLIN, AGLOGLINYGRID | AGLOGLINSYMBOLS | AGLOGLINSOLID | AGLOGLINTHICK, 0 );

    AGClose();
}
```

## Pie chart

```
void CGSMainWnd::GraphPie2D()
{
    char    FAR* strTitleg = " regional sales distribution";
    char    FAR* strTitlex = "1992";

    char    strTitle [60];
    double  fSales   [NUMMONTHS];
    int     Aux      [NUMMONTHS];
    int     MonthNum, RegionNum;

    RegionNum = rand() % NUMREGIONS;
    lstrcpy( strTitle, g_lpstrRegionNames[RegionNum] );
    lstrcat( strTitle, strTitleg );

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fSales[MonthNum] = g_fSales[MonthNum][RegionNum];
        Aux[MonthNum] = MonthNum == 4 ? PCEXPL : FALSE;
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fSales );
    AGAux( NUMMONTHS, Aux );
    AGLegend( NUMMONTHS, g_lpstrMonthNames );
    AGTitleG( strTitle );
    AGTitleX( strTitlex );
    SelectFonts();

    AGShow( AGPIE2D, AGPIEPERCENT | AGPIEPCCHAR, 0 );

    AGClose();
}
```



## 3D pie chart

```
void CGSMainWnd::GraphPie3D()
{
    char    FAR* strTitleg = " department budget allocation";
    char    FAR* strTitlex = "1992";

    char    strTitle [60];
    double  fCosts   [NUMMONTHS];
    int     Aux       [NUMMONTHS];
    int     MonthNum, RegionNum;

    RegionNum = rand() % NUMREGIONS;
    lstrcpy( strTitle, g_lpstrDeptNames[RegionNum] );
    lstrcat( strTitle, strTitleg );

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fCosts[MonthNum] = g_fCosts[MonthNum][RegionNum];
        Aux[MonthNum] = MonthNum == 4 ? PCEXPL : FALSE;
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fCosts );
    AGAux( NUMMONTHS, Aux );
    AGLegend( NUMMONTHS, g_lpstrMonthNames );
    AGTitleG( strTitle );
    AGTitleX( strTitlex );
    SelectFonts();

    AGShow( AGPIE3D, AGPIEPERCENT | AGPIEPCCHAR, 0 );

    AGClose();
}
```

## Polar graph

```
void CGSMainWnd::GraphPolar()
{
    char    strTitleg [60], FAR* pstrTitleg = "Arctic roll sales";
    char    strTitlex [20], FAR* pstrTitlex = "1992";

    OemToAnsi( pstrTitleg, strTitleg );
    OemToAnsi( pstrTitlex, strTitlex );

    int      Patt      [1];
    int      Symbol    [1];
    int      RegionNum;

    for ( RegionNum = 0; RegionNum < 1; ++RegionNum ) {
        Patt[RegionNum] = 2; Symbol[RegionNum] = 3 + 2 * RegionNum;
    }

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    AGOpen();
    AGAmp( NUMMONTHS, 1, &g_fSales[0][0] );
    AGPatt( 1, Patt );
    AGSym( 1, Symbol );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( 1, g_lpstrRegionNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    SelectFonts();

    AGShow( AGPOLAR,
            AGPOLARNOLABELS | AGPOLARANGGRID | AGPOLARSYMBOL | AGPOLARLINE | AGPOLARTHICK,
            0 );

    m_GraphAxisLowVal = 0;
    AGClose();
}
```

## Graph statistics

```
void CGSMainWnd::GraphStatistics()
{
    char    FAR* strTitleg = " regional profitability statistics";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    char    strTitle [60];
    double  fSales   [NUMMONTHS];
    int     Patt     [1];
    int     Symbol   [1];
    int     MonthNum, RegionNum;

    RegionNum = rand() % NUMREGIONS;
    strcpy( strTitle, g_lpstrRegionNames[RegionNum] );
    strcat( strTitle, strTitleg );

    for ( MonthNum = 0; MonthNum < NUMMONTHS; ++MonthNum ) {
        fSales[MonthNum] = g_fSales[MonthNum][RegionNum];
    }

    Patt[0] = 2; Symbol[0] = 3 + 2 * RegionNum;

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );
    GSSizeSymbol( 0.03 * GSGetVYExt() );

    AGOpen();
    AGAmp( NUMMONTHS, 1, fSales );
    AGPatt( 1, Patt );
    AGSym( 1, Symbol );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGTitleG( strTitle );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    AGYAxisStyle( AGVARORIGIN, 0, 0, 0, 0 );
    SelectFonts();

    AGShow( AGLINE, AGLINESYMBOLS, AGMEAN | AGMINMAX | AGSD );

    AGClose();
}
```

## Tape graph

```
void CGSMainWnd::GraphTape()
{
    char    FAR* strTitleg = "Departmental costs";
    char    FAR* strTitlex = "1992";
    char    FAR* strTitley = "$";

    GSSetBG( m_StyleBG );
    GSClearView( CLOPAQUE );

    AGOpen();
    AGamp( NUMMONTHS, NUMREGIONS, &g_fCosts[0][0] );
    AGLabels( NUMMONTHS, g_lpstrMonthLabels );
    AGLegend( NUMREGIONS, g_lpstrDeptNames );
    AGTitleG( strTitleg );
    AGTitleX( strTitlex );
    AGTitleY( strTitley );
    SelectFonts();

    AGShow( AGTAPE, 0, 0 );

    AGClose();
}
```

## Custom graph example

This is an example of a graph drawn using the custom graph functions of Graphics Server. It shows some of the extra styling made possible by calling the custom graph functions directly. The code which produced this graph is not part of this application. The graph was drawn by another application and then the image was exported by GSPicWrite. The task of the function in this application is simply to re-import the image with GSPicRead.

```
void CGSMainWnd::GraphExample1()
{
    GSClearView( CLTRANSP );
    GSPicRead( 0, 0, 0, 0, PXPMPF, PXSTRETCH, "GSWEXAM1.WMF" );
}
```

## Custom graph example

This is an example of a graph drawn using the custom graph functions of Graphics Server. It shows some of the extra styling made possible by calling the custom graph functions directly. The code which produced this graph is not part of this application. The graph was drawn by another application and then the image was exported by GSPicWrite. The task of the function in this application is simply to re-import the image with GSPicRead.

```
void CGSMainWnd::GraphExample2()
{
    GSClearView( CLTRANSP );
    GSPicRead( 0, 0, 0, 0, PXPMPF, PXSTRETCH, "GSWEXAM2.WMF" );
}
```

## Custom graph example

This is an example of a graph drawn using the custom graph functions of Graphics Server. It shows some of the extra styling made possible by calling the custom graph functions directly. The code which produced this graph is not part of this application. The graph was drawn by another application and then the image was exported by GSPicWrite. The task of the function in this application is simply to re-import the image with GSPicRead.

```
void CGSMainWnd::GraphExample3()
{
    GSClearView( CLTRANSP );
    GSPicRead( 0, 0, 0, 0, PXPMPF, PXSTRETCH, "GSWEXAM3.WMF" );
}
```

## **Availability**

Graphics Server is a product of Bits Per Second Ltd, a world leader in graphical utilities for PC's.

For further details and ordering information please call:

Pinnacle Publishing Inc.  
P.O. Box 888  
Kent, WA 98035-0888

Telephone: 800/788-1900  
206/251-1900  
FAX: 206/251-5057

Graphics Server run-time files may be issued with your applications licence-free.