Olectra Chart™

3D DLL
Programmer's Guide & Reference Manual

Version 1.1

The Leader in GUI Components

260 King Street East
Toronto, Ontario, Canada M5A 1K3
(416) 594-1026
www.klg.com

June 1996

RefNo: 3DLLPR-CH/11-06/96

Printed in Canada on recycled paper. 

Table of Contents

Preface 1
Introduction 1
Assumptions 2
Typographical Conventions Used in This Manual 2
Overview of Manual 2
Related Documents 3

Part I: Using the Chart

1 Getting Started: Developing a Simple Olectra Chart Program . . 7
1.1 Introduction 7
1.2 The SIMPLE.C Program 7
1.3 The SIMPLE.C 3D Data 8
1.4 The SIMPLE.C Controls 9

2 Olectra Chart Basics 11
2.1 Terminology 11
2.2 Property Setting and Retrieving 12
2.3 USE_DEFAULT Properties 13
2.4 Pointer Properties 14
2.5 Batching Property Updates 15
2.6 Font Properties 15
2.7 Programming with C++ 15
2.8 Distributing Olectra Chart Applications 16

3 Programming Olectra Chart 19
3.1 Properties 19
3.2 15 Basic Types of Surfaces and Bars 24
3.3 Bar Charts and Histograms 29
3.4 Contours and Zone Display 31
3.5 Axis Controls 33
3.6 Legend Display 34

3.7 Perspective 35
3.8 Mesh Controls 36
3.9 Surface Colors 37
3.10 Solid Surface 37
3.11 Axis Scaling 38
3.12 Axis Labelling 38
3.13 Grid Lines 39
3.14 Header and Footer Text 40
3.15 Header, Footer and Legend Fonts 40
3.16 Area Positioning 41
3.17 Area Borders 42
3.18 Foreground and Background Colors 43
3.19 Double Buffering 44
3.20 Output and Printing 44

4 Olectra Chart Data 47
4.1 Data Overview 47
4.2 The Xrt3dData Structure 49

5 Programming User Interaction 53
5.1 Default User Interaction 53
5.2 Overview of Action Maps and Messages 55
5.3 Starting User Interaction 55
5.4 Updating User Interaction 56
Scaling, Translation, and Zooming 56
Rotation 57
5.5 Ending User Interaction 58
5.6 Programming Actions 58
Changing the Action Maps 58
Disabling and Disallowing Interactions 60
Calling Actions Directly 60
5.7 Interacting with Chart Data 61
5.8 Window Resizing 63

6 Advanced Olectra Chart Programming 65
6.1 4D Surface Charts 65
6.2 4D Bar Charts 66
6.3 Text Objects 67
6.4 Customizing the Distribution Table 70
6.5 Customizing Legend Labels 71
6.6 Customizing Contour Styles 73

Part II: Reference Appendices

A Olectra Chart Property Reference 79

A.1 Control Synopsis 79

A.2 Olectra Chart Properties 79

B Olectra Chart Text Object Property Reference. 95

C Procedures and Methods Reference 99

D Message Reference. 121

E Data Types 125

F Sample Code 135

F.1 SIMPLE.C 136

Preface

Introduction ■ *Assumptions*
Typographical Conventions Used in This Manual ■ *Related Documents*

Introduction

The Olectra Chart Control

Windows contains many different pre-defined controls such as buttons, scrollbars, and list boxes. Conceptually, Olectra Chart adds a new control class to Windows. The control displays 3D data in surface, contour, and bar chart representations.

Olectra Chart has properties that determine how the chart will look and behave. Writing programs using Olectra Chart is very similar to writing any other kind of Windows program; you now have one more Windows control to use.

Olectra Chart has properties which allow control of:

- Drawing (whether to draw the mesh, surface, contours and/or zones) and chart type (surface or bar chart).
- Contour styles: line widths, colors and patterns, and fill colors.
- Distribution method, distribution table, number of distribution levels.
- Hidden-line display, surface colors and mesh colors.
- 3D rotation and perspective.
- Legend positioning, orientation, style, border style, anchor, font and color.
- Header and footer positioning, border style, text, font and color.
- Chart positioning, border style, color, width, and height.
- Axis maximums and minimums, fonts, gridlines, titles and labels.
- Window background and foreground color.

Olectra Chart also provides several procedures and methods which:

- Allocate and load data structures containing the numbers to be graphed.
- Assist the developer in dealing with user-events.
- Assist the developer in dealing with interactive rotation, zooming and shifting of the 3D view by the end-user.
- Assist the developer with setting and getting indexed properties.

Assumptions

This manual assumes that the reader is proficient with the C language and the Windows API. C concepts such as "an array of **char ***" and "pointer to a structure" must be understood before beginning to program Windows and Olectra Chart applications. An understanding of basic Windows programming concepts such as event-driven programming and programming Windows controls is required before continuing with this manual. See page 3 for information on Windows programming references.

Typographical Conventions Used in This Manual

Bold	■ Chart property and method names. ■ Language-specific keywords, constants, variables, and function names. ■ Commands that you enter at a command prompt.
Italic Text	■ Parameter names and information you specify. ■ New terms as they are introduced, and to emphasize important words. ■ Figure and table titles. ■ The names of other documents referenced in this manual, such as the *Getting Started with Olectra Chart* booklet.
UPPERCASE	■ File and directory names, key names, and key sequences.
`Monospace`	■ Code examples, variables in body text, and error text.
...[XYZ]...	Many Olectra Chart properties used to specify axis information are similar for the X-, Y-, and Z-axis. The syntax **[XYZ]** used in an property name refers to one of all of the X, Y, or Z versions of this property. For example, "**XRT3D_XMIN**, **XRT3D_YMIN**, and **XRT3D_ZMIN**" is shortened to "**XRT3D_[XYZ]MIN**".

Overview of Manual

Part I describes how to use Olectra Chart.

Chapter 1, "Getting Started: Developing a Simple Olectra Chart Program", provides a hands-on introduction to Olectra Chart by giving you the chance to compile and run a simple example program. It should be read by all programmers learning to use Olectra Chart.

Chapter 2, "Olectra Chart Basics", provides basic information that you need to know before starting to develop applications with Olectra Chart. It gives terminology, an overview of the control, and provides some programming approaches that apply to many properties.

Chapter 3, “Programming Olectra Chart”, provides programming information for most Olectra Chart properties.

Chapter 4, “Olectra Chart Data”, provides details on getting data into charts and manipulating the **Xrt3dData** structure.

Chapter 5, “Programming User Interaction”, provides programming and end-user information on Olectra Chart’s user interaction capabilities.

Chapter 6, “Advanced Olectra Chart Programming”, discusses more advanced and less commonly-used aspects of Olectra Chart, including: creating 4D charts; using text objects; creating charts with custom distribution tables; custom legend labels; and customizing contour styles.

Part II contains detailed technical reference information in a number of appendices.

Appendix A, “Olectra Chart Property Reference”, provides a concise reference of all Olectra Chart properties in alphabetical order.

Appendix B, “Olectra Chart Text Object Property Reference”, describes all Olectra Chart text object properties.

Appendix C, “Procedures and Methods Reference”, describes all Olectra Chart procedures and methods in alphabetical order.

Appendix D, “Message Reference”, provides a reference of Olectra Chart’s notification messages.

Appendix E, “Data Types”, lists the Olectra Chart data structures.

Appendix F, “Sample Code”, provides an overview of the sample programs included with Olectra Chart, and lists two short example programs.

Related Documents

The following documents are useful references for Windows application development:

- *Programming Windows 3.1* by Charles Petzold, Microsoft Press.
- *Windows 3.1 SDK* and *Windows 3.1 API* online documentation, Microsoft Corporation.
- *Advanced Windows Programming* by Jeffrey Richter, Microsoft Press.
- *Win32 SDK* and *Win32 API* online documentation, Microsoft Corporation.

Part I

Using the Chart

1

Getting Started: Developing a Simple Olectra Chart Program

Introduction ■ *The SIMPLE.C Program*
The SIMPLE.C 3D Data ■ *The SIMPLE.C Controls*

1.1 Introduction

This chapter allows you to immediately try out Olectra Chart by compiling and running an example application, introducing some fundamental Olectra Chart concepts. The application, SIMPLE.C, graphs a basic mathematical function and allows a user to directly manipulate the four most important Olectra Chart drawing controls: DrawMesh, DrawShaded, DrawContours, and DrawZones.

This program is listed in Appendix F and can be found online in Olectra Chart's \CHART\3D\DEMOS\DLL\SDK\SIMPLE directory.

1.2 The SIMPLE.C Program

In this chapter we'll describe an introductory Olectra Chart program called SIMPLE.C. A complete listing of this program can be found in Olectra Chart's \CHART\3D\DEMOS\DLL\SDK\SIMPLE directory. SIMPLE.C is listed in Appendix F on page 135.

When SIMPLE.C is compiled and run, the window shown in Figure 1 is displayed. Try compiling and running SIMPLE.C before continuing with this section.

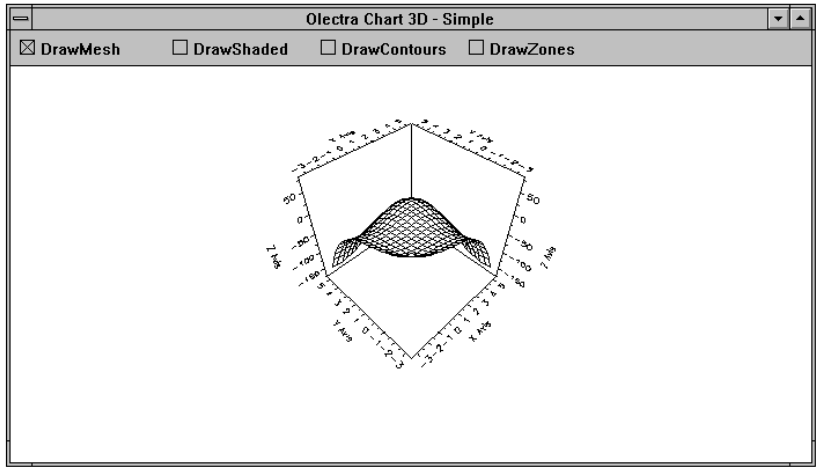


Figure 1 The Window Created by SIMPLE.C

SIMPLE.C illustrates several Olectra Chart programming principles. It creates a window containing a row of check-box controls. Below the check-box controls, a chart control is created. If the window is resized, the chart control is automatically resized.

1.3 The SIMPLE.C 3D Data

A common task in any Olectra Chart program is to load the data to be graphed into an **Xrt3dData** structure. The data can come from any source: a file or database, another process running on the machine, or can be calculated on the fly. SIMPLE.C creates an **Xrt3dData** structure and loads it with grid data from the surface defined by the equation:

$$z = 3xy - x^3 - y^3 \quad -3 \le x, y \le 5$$

Chapter 4 on page 47 provides complete details on Olectra Chart data, but for now, think of a grid aligned with the X and Y axis projected onto the surface we've defined above. When an X grid line crosses a Y grid line, the intersection defines a point on the surface. We'll evaluate the surface equation at that point, and pass the solution into the chart control for graphing.

The first decision to make is: How many grid points? Too few grid points will result in too few samples, and a low resolution image. Too many will result in extra overhead, without adding clarity to the on-screen display. Olectra Chart is optimized for grid sizes less than 100 x 100.

SIMPLE.C defines the **Xrt3dData** structure in the **CalculateGrid**() routine. **nRows** and **nColumns** are both assigned the value 20. They will define the grid resolution.

SIMPLE.C calls **Xrt3dMakeGridData**() to allocate an **Xrt3dData** structure with enough space to hold 20 x 20 Z-values:

```
Xrt3dGridData* pGridData = (Xrt3dGridData*)
    Xrt3dMakeGridData(nRows, nColumns, XRT3D_HUGE_VAL,
        8.0 / (nRows - 1), 8.0 / (nColumns - 1),
        -3.0, -3.0, TRUE);
```

If **Xrt3dMakeGridData**() is successful, it returns a pointer to an **Xrt3dData** structure. If it is not successful, it returns **NULL**. The first two parameters, **nRows** and **nColumns**, tell it how large a structure to allocate.

The third parameter indicates the data value that will be used to mean "no value". Since there are no holes in our surface data, we set this to **XRT3D_HUGE_VAL**, a value which is unlikely to appear in our data values.

The next two parameters specify the step distance between grid lines in the X and Y directions. Since our grid will have to span from -3 to 5 using 20 grid lines, the distance between grid lines will be 8.0 / (20 - 1) or about 0.421.

The next two parameters indicate the location of our data's origin (-3, -3). Finally, the last parameter tells **Xrt3dMakeGridData**() whether or not to allocate space for the Z values. We do not want to allocate this ourselves, so we set it to **TRUE**.

Now that we have an **Xrt3dData** structure to hold our grid data, we can go ahead and evaluate the surface equation at each of the grid points. The *g.values* element in the **Xrt3dData** structure is a two dimensional array of type **double**. The following code populates it with our surface's grid values:

```
for (i = 0; i < nRows; i++) {
    x = pGridData->xorig + i * pGridData->xstep;
    for (j = 0; j < nColumns; j++) {
        y = pGridData->yorig + j * pGridData->ystep;
        pGridData->values[i][j] = 3*x*y - x*x*x - y*y*y;
    }
}
```

Our **Xrt3dData** structure referenced by grid is now ready for graphing. When SIMPLE.C creates the chart control, it attaches the data to the control by setting the **XRT3D_SURFACE_DATA** property to **grid**.

1.4 The SIMPLE.C Controls

At the top of the SIMPLE.C application window are a number of toggles that the user can adjust.

The DrawMesh, DrawShaded, DrawContours and DrawZones toggles turn the corresponding **XRT3D_DRAW_MESH**, **XRT3D_DRAW_SHADED**, **XRT3D_DRAW_CONTOURS** and **XRT3D_DRAW_ZONES** Boolean properties on and off.

These four properties provide 15 different combinations of basic 3D data display. Some combinations will be more useful for your applications than others. These properties are described in section 3.2 on page 24.

Most Olectra Chart properties are discussed in the following sections, and all are fully described in Appendix A.

2

Olectra Chart Basics

Terminology ■ *Property Setting and Retrieving*
USE_DEFAULT Properties ■ *Pointer Properties*
Batching Property Updates ■ *Font Properties*
Programming with C++ ■ *Distributing Olectra Chart Applications*

Conceptually, Olectra Chart adds a new control to the pre-defined Windows control classes (such as Button, ScrollBar, and ListBox). A Windows programmer manipulates this control similarly to other controls.

2.1 Terminology

Figure 2 shows the major components of the chart control. There are four major areas: Header, Footer, Legend and Graph. Each area has its own origin, width and height.

Unit Cube

The *unit cube* is defined to be the smallest cube which encloses the entire 3D scene (including the axis). Some Olectra Chart properties, such as those that specify axis scaling and axis font sizes, have definitions that depend on the unit cube size.

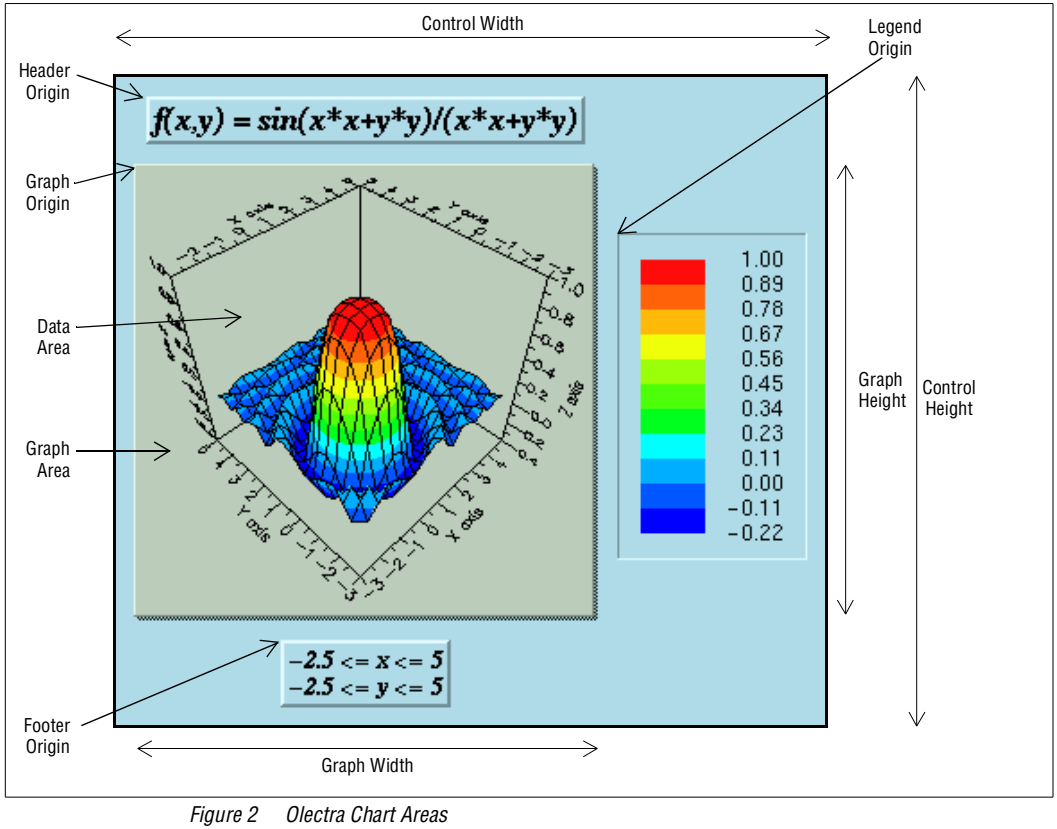


Figure 2 Olectra Chart Areas

2.2 Property Setting and Retrieving

To set the value of chart properties, use either the **Xrt3dSetValues**() or **Xrt3dSetPropString**() procedure. To retrieve the current value of chart properties, use either the **Xrt3dGetValues**() or **Xrt3dGetPropString**() procedure.

SetValues / GetValues

Xrt3dSetValues() and **Xrt3dGetValues**() allow you to set or get any number of properties at once. The following example sets two chart properties to the values contained in the **bg** and **fg** variables:

```
Xrt3dSetValues(hChart,
    XRT3D_BACKGROUND_COLOR,bg,
    XRT3D_FOREGROUND_COLOR,fg,
    NULL);
```

To retrieve values using **Xrt3dGetValues**(), pass the address of the variable to contain the value retrieved. For example:

```
Xrt3dGetValues(hChart,
    XRT3D_BACKGROUND_COLOR,&bg,
    XRT3D_FOREGROUND_COLOR,&fg,
    NULL);
```

Make sure that each variable is of the proper type before calling **Xrt3dGetValues**(). The resource value types are listed at the start of Chapter 3.

SetPropString / GetPropString

Xrt3dSetPropString() and **Xrt3dGetPropString**() allow you to set or get a property value as a string. This is particularly useful for setting fonts, colors, and datastyles. Olectra Chart converts a string to/from the data type of the property. The following example sets the header font:

```
Xrt3dSetPropString(hChart,
    XRT3D_HEADER_FONT,"Times,10,Bold");
```

To retrieve a value using **Xrt3dGetPropString**(), pass the address of the variable to contain the value retrieved. For example:

```
char * my_font;
Xrt3dGetPropString(hChart,
    XRT3D_HEADER_FONT,&my_font);
...
Xrt3dFreePropString(my_font);
```

Use **Xrt3dFreePropString**() to free the string allocated by **Xrt3dGetPropString**() after use.

2.3 USE_DEFAULT Properties

Many Olectra Chart properties have corresponding **USE_DEFAULT** properties, for example, **XRT3D_HEADER_X** and **XRT3D_HEADER_X_USE_DEFAULT**.

USE_DEFAULT properties are Booleans that determine if Olectra Chart should calculate a default value for the property.

For example, if **XRT3D_HEADER_X_USE_DEFAULT** is **TRUE**, every time the Olectra Chart window is resized, Olectra Chart will determine a reasonable default value for the X pixel coordinate for the header area position. Otherwise, Olectra Chart will use the last specified value of **XRT3D_HEADER_X** to position the header area.

Attempts to set a **USE_DEFAULT** property to **FALSE** are ignored by Olectra Chart unless the corresponding property has been explicitly set or previously calculated by Olectra Chart. A side effect of setting any property that has a corresponding **USE_DEFAULT** property is that the **USE_DEFAULT** property will be set to **FALSE**.

The following code will freeze the value of **XRT3D_XMAX** at its current value. It will also have the side effect of setting **XRT3D_XMAX_USE_DEFAULT** to **FALSE**.

```
double  xmax;
Xrt3dGetValues(mygraph,
    XRT3D_XMAX, &xmax, NULL);
Xrt3dSetValues(mygraph, XRT3D_XMAX, xmax, NULL);
```

The following code will revert back to the default behavior, enabling Olectra Chart to calculate a default value for **XRT3D_XMAX** whenever it draws the chart.

```
Xrt3dSetValues(mygraph,
    XRT3D_XMAX_USE_DEFAULT,  TRUE,
    NULL);
```

2.4 Pointer Properties

Some Olectra Chart properties return pointers to structures or characters when used in an **Xrt3dGetValues**() call. A program should never use this pointer to change any data in memory. Everything the pointer points to should be considered "read-only".

The only exceptions to this rule are the **XRT3D_SURFACE_DATA** and **XRT3D_ZONE_DATA** properties. These properties should always point to data structures that exist in program memory.

Olectra Chart makes its own copy of all data passed through a pointer property (except for **XRT3D_SURFACE_DATA** and **XRT3D_ZONE_DATA**).

The "duplicating" procedures **Xrt3dDupContourStyles**(), **Xrt3dDupDistnTable**(), **Xrt3dDupStrings**(), **Xrt3dDupValueLabels**(), and **Xrt3dDupXYColors**() can be used to make copies of some types of data. These copies can be freed with corresponding "free" procedures (e.g. **Xrt3dFreeStrings**()). See Appendix C on page 99 for a complete list of procedures.

For example, if a program wants to interchange the first header string with the first footer string, the following code could be used:

```
char    **hs, **fs, **myhs, **myfs, *tmp;
Xrt3dGetValues(mygraph,         /* Get read-only ptrs */
    XRT3D_HEADER_STRINGS, &hs,
    XRT3D_FOOTER_STRINGS, &fs
    NULL);

myhs = Xrt3dDupStrings(hs);     /* Make copies */
myfs = Xrt3dDupStrings(fs);

tmp = myfs[0];                  /* Interchange 1st */
myfs[0] = myhs[0];
myhs[0] = tmp;

Xrt3dSetValues(mygraph,         /* Reset them */
    XRT3D_HEADER_STRINGS, myhs,
    XRT3D_FOOTER_STRINGS, myfs,
    NULL);

Xrt3dFreeStrings(myhs);         /* Free my copies */
Xrt3dFreeStrings(myfs);
```

2.5 Batching Property Updates

Normally property changes take effect immediately after the **Xrt3dSetValues()** call. If you would prefer to make several changes to the control's properties before causing a repaint, set **XRT3D_REPAINT** to **FALSE**. All property changes will be batched until **XRT3D_REPAINT** is set to **TRUE**.

2.6 Font Properties

If you are using Microsoft Windows 3.1, you should always cast the property value to an **int** when setting font properties. For example:

```
HFONT   hfont;

Xrt3dSetValues(hChart, XRT3D_HEADER_FONT, (int)hfont, NULL);
```

2.7 Programming with C++

Olectra Chart provides two C++ interfaces to the control:

1. MFC – For use within version 2.0 or greater of MFC, subclassed from the **CWnd** visual object class.
2. OWL – For use within version 2.5 or greater of OWL, subclassed from the **TControl** ObjectWindows class.

To use the MFC classes, include the OCH3DMFC.H header file and OCH3DMFC.CPP source file in your application. Both files are located in Olectra Chart's \INCLUDE directory. The \CHART\3D\DEMOS\DLL\MFC directory contains a simple MFC example program.

To use the OWL classes, include the OCH3DOWL.H header file and OCH3DOWL.CPP source file in your application. Both files are located in Olectra Chart's \INCLUDE directory. The \CHART\3D\DEMOS\DLL\OWL directory contains a simple OWL example program.

The MFC and OWL implementations create the following classes:

MFC Class	OWL Class	Methods
CChart3d	TChart3d	■ A constructor that creates the chart object. ■ A destructor that calls **DestroyWindow**(). ■ A method for each of the C procedures that take an Olectra Chart control as its first argument. For example, the method for **Xrt3dCallAction**() is **CallAction**(). ■ A method to set each settable property. For example, the method to set **XRT3D_BACKGROUND_COLOR** is **SetBackgroundColor**(). ■ A method to retrieve each property. For example, the method to retrieve **XRT3D_BACKGROUND_COLOR** is **GetBackgroundColor**().
CChart3dData	TChart3dData	■ A constructor that calls one of **Xrt3dMakeData**(), **Xrt3dMakeDataFromFile**(), or **Xrt3dDataCopy**(). ■ A destructor that calls **Xrt3dDestroyData**(). ■ A method for most of the C procedures that take an **Xrt3dData** structure as its first argument. For example, the method for **Xrt3dDataSmooth**() is **DataSmooth**(). ■ Inline methods for **Xrt3dData** macros. ■ Overloaded = operator to perform **Xrt3dDataCopy**(). ■ Casting support for (**Xrt3dData ***).

2.8 Distributing Olectra Chart Applications

You can freely distribute any applications that you create with Olectra Chart. An Olectra Chart application needs its dynamic link library present on the system it is run on.

Distributing 32-bit Applications

When distributing 32-bit applications, you may only distribute the OLCH3D32.DLL dynamic link library.

Distributing 16-bit Applications

When distributing 16-bit applications, you may only distribute the OLCH3D16.DLL dynamic link library.

3

Programming Olectra Chart

Properties ■ *15 Basic Types of Surfaces and Bars*
Bar Charts and Histograms ■ *Contours and Zone Display*
Axis Controls ■ *Legend Display*
Perspective ■ *Mesh Controls*
Surface Colors ■ *Solid Surface*
Axis Scaling ■ *Axis Labelling*
Grid Lines ■ *Header and Footer Text*
Header, Footer and Legend Fonts ■ *Area Positioning*
Area Borders ■ *Foreground and Background Colors*
Double Buffering ■ *Output and Printing*

3.1 Properties

This section summarizes all of the Olectra Chart properties. It is not necessary to remember all the properties in order to program Olectra Chart effectively. For most charts, many properties may be left with their default settings.

This section will not describe each property in detail. Most properties are described in this chapter, and each property is summarized in Appendix A. Scan through these tables to gather a basic understanding of the properties.

Property	Type
XRT3D_CONTOUR_STYLES	Xrt3dContourStyle **
XRT3D_DATA_AREA_BACKGROUND_COLOR	COLORREF
XRT3D_DISTN_METHOD	Xrt3dDistnMethod
XRT3D_DISTN_TABLE	Xrt3dDistnTable *
XRT3D_DRAW_MESH	BOOL
XRT3D_DRAW_SHADED	BOOL
XRT3D_DRAW_CONTOURS	BOOL
XRT3D_DRAW_ZONES	BOOL
XRT3D_DRAW_HIDDEN_LINES	BOOL
XRT3D_MESH_BOTTOM_COLOR	COLORREF
XRT3D_MESH_TOP_COLOR	COLORREF
XRT3D_NUM_DISTN_LEVELS	int
XRT3D_PERSPECTIVE_DEPTH	double
XRT3D_PROJECT_ZMAX	int
XRT3D_PROJECT_ZMIN	int
XRT3D_SOLID_SURFACE	BOOL
XRT3D_SURFACE_BOTTOM_COLOR	COLORREF
XRT3D_SURFACE_TOP_COLOR	COLORREF
XRT3D_SURFACE_DATA	Xrt3dData *
XRT3D_TYPE	Xrt3dType
XRT3D_VIEW_NORMALIZED	BOOL
XRT3D_VIEW_SCALE	double
XRT3D_VIEW_[XY]TRANSLATE	double
XRT3D_[XY]MESH_FILTER	int
XRT3D_[XY]MESH_SHOW	BOOL
XRT3D_[XYZ]ROTATION	double
XRT3D_[XYZ]SCALE	double
XRT3D_ZONE_DATA	Xrt3dData *
XRT3D_ZONE_METHOD	Xrt3dZoneMethod

Figure 3 Data Display Properties

Property	Type
XRT3D_GRAPH_BACKGROUND_COLOR	COLORREF
XRT3D_GRAPH_BORDER	Xrt3dBorder
XRT3D_GRAPH_BORDER_WIDTH	int
XRT3D_GRAPH_FOREGROUND_COLOR	COLORREF
XRT3D_GRAPH_HEIGHT	int
XRT3D_GRAPH_HEIGHT_USE_DEFAULT	BOOL
XRT3D_GRAPH_WIDTH	int
XRT3D_GRAPH_WIDTH_USE_DEFAULT	BOOL
XRT3D_GRAPH_[XY]	int
XRT3D_GRAPH_[XY]_USE_DEFAULT	BOOL

Figure 4 Chart Properties

Property	Type
XRT3D_AXIS_STROKE_FONT	Xrt3dStrokeFont
XRT3D_AXIS_STROKE_SIZE	int
XRT3D_AXIS_TITLE_STROKE_FONT	Xrt3dStrokeFont
XRT3D_AXIS_TITLE_STROKE_SIZE	int
XRT3D_[XY]DATA_LABELS	char **
XRT3D_[XYZ]ANNO_METHOD	Xrt3dAnnoMethod
XRT3D_[XYZ]AXIS_SHOW	BOOL
XRT3D_[XYZ]AXIS_TITLE	char *
XRT3D_[XYZ]GRID_LINES	int
XRT3D_[XYZ]MAX	double
XRT3D_[XYZ]MAX_USE_DEFAULT	BOOL
XRT3D_[XYZ]MIN	double
XRT3D_[XYZ]MIN_USE_DEFAULT	BOOL
XRT3D_[XYZ]VALUE_LABELS	Xrt3dValueLabel **
XRT3D_ZORIGIN	double

Figure 5 Axis Properties

Property	Type
XRT3D_HEADER_ADJUST	Xrt3dAdjust
XRT3D_HEADER_BACKGROUND_COLOR	COLORREF
XRT3D_HEADER_BORDER	Xrt3dBorder
XRT3D_HEADER_BORDER_WIDTH	int
XRT3D_HEADER_FOREGROUND_COLOR	COLORREF
XRT3D_HEADER_STRINGS	char **
XRT3D_HEADER_FONT	HFONT
XRT3D_HEADER_HEIGHT	int
XRT3D_HEADER_WIDTH	int
XRT3D_HEADER_[XY]	int
XRT3D_HEADER_[XY]_USE_DEFAULT	BOOL
XRT3D_FOOTER_ADJUST	Xrt3dAdjust
XRT3D_FOOTER_BACKGROUND_COLOR	COLORREF
XRT3D_FOOTER_BORDER	Xrt3dBorder
XRT3D_FOOTER_BORDER_WIDTH	int
XRT3D_FOOTER_FONT	HFONT
XRT3D_FOOTER_FOREGROUND_COLOR	COLORREF
XRT3D_FOOTER_HEIGHT	int
XRT3D_FOOTER_STRINGS	char **
XRT3D_FOOTER_WIDTH	int
XRT3D_FOOTER_[XY]	int
XRT3D_FOOTER_[XY]_USE_DEFAULT	BOOL

Figure 6 Header and Footer Properties

Property	Type
XRT3D_[XY]BAR_FORMAT	Xrt3dBarFormat
XRT3D_[XY]BAR_SPACING	double
XRT3D_XY_COLORS	Xrt3dXYColor **

Figure 7 Bar Properties

Property	Type
XRT3D_LEGEND_ANCHOR	Xrt3dAnchor
XRT3D_LEGEND_BACKGROUND_COLOR	COLORREF
XRT3D_LEGEND_BORDER	Xrt3dBorder
XRT3D_LEGEND_BORDER_WIDTH	int
XRT3D_LEGEND_FONT	HFONT
XRT3D_LEGEND_FOREGROUND_COLOR	COLORREF
XRT3D_LEGEND_HEIGHT	int
XRT3D_LEGEND_LABEL_FUNC	Function
XRT3D_LEGEND_ORIENTATION	Xrt3dAlign
XRT3D_LEGEND_SHOW	BOOL
XRT3D_LEGEND_STRINGS	char **
XRT3D_LEGEND_STYLE	Xrt3dLegendStyle
XRT3D_LEGEND_WIDTH	int
XRT3D_LEGEND_[XY]	int
XRT3D_LEGEND_[XY]_USE_DEFAULT	BOOL

Figure 8 Legend Properties

Property	Type
XRT3D_BACKGROUND_COLOR	COLORREF
XRT3D_BORDER	Xrt3dBorder
XRT3D_BORDER_WIDTH	int
XRT3D_DEBUG	BOOL
XRT3D_DOUBLE_BUFFER	BOOL
XRT3D_FOREGROUND_COLOR	COLORREF
XRT3D_HEIGHT	int
XRT3D_NAME	char *
XRT3D_REPAINT	BOOL
XRT3D_WIDTH	int

Figure 9 Other Properties

Property	Type
XRT3D_TEXT_ADJUST	Xrt3dAdjust
XRT3D_TEXT_ATTACH_INDEX_[XY]	int
XRT3D_TEXT_ATTACH_METHOD	Xrt3dTextAttachMethod
XRT3D_TEXT_ATTACH_PIXEL_[XY]	int
XRT3D_TEXT_ATTACH_POINT_[XYZ]	double
XRT3D_TEXT_BACKGROUND_COLOR	COLORREF
XRT3D_TEXT_BORDER	Xrt3dBorder
XRT3D_TEXT_BORDER_WIDTH	int
XRT3D_TEXT_FONT	HFONT
XRT3D_TEXT_FOREGROUND_COLOR	COLORREF
XRT3D_TEXT_LINE_SHOW	BOOL
XRT3D_TEXT_OFFSET_[XY]	int
XRT3D_TEXT_PLANE	int
XRT3D_TEXT_PRINT_FONT	char *
XRT3D_TEXT_SHOW	BOOL
XRT3D_TEXT_STRINGS	char **
XRT3D_TEXT_STROKE_FONT	Xrt3dStrokeFont
XRT3D_TEXT_STROKE_SIZE	int

Figure 10 Text Object Properties

3.2 15 Basic Types of Surfaces and Bars

Any data that has been attached to the chart control using the **XRT3D_SURFACE_DATA** property will be displayed in either a surface or bar representation. The type of representation depends on the value of the **XRT3D_TYPE** property, which can be either **XRT3D_TYPE_SURFACE** or **XRT3D_TYPE_BAR**.

Olectra Chart's four basic display boolean properties–DrawMesh, DrawShaded, DrawContours and DrawZones–combine to create 15 different basic surface and bar displays.[1]

DrawMesh

Surfaces: When DrawMesh is **TRUE,** Olectra Chart displays the X-Y grid projected onto the 3D surface in a 3D view with a Z-axis. The chart honors rotation and perspective control. The **XRT3D_DRAW_HIDDEN_LINES**, **XRT3D_[XY]MESH_SHOW** and **XRT3D_[XY]MESH_FILTER** properties are used to fine-tune the mesh display, and **XRT3D_MESH_BOTTOM_COLOR** and **XRT3D_MESH_TOP_COLOR** properties control the mesh colors.

1. No chart is displayed when all four booleans are FALSE.

Bars: When DrawMesh is **TRUE**, Olectra Chart will draw the outline of all the bars. All bars with value greater than or equal to **XRT3D_ZORIGIN** will be outlined using **XRT3D_MESH_TOP_COLOR**, and all bars with value less than **XRT3D_ZORIGIN** will be outlined using **XRT3D_MESH_BOTTOM_COLOR**. When **XRT3D_DRAW_HIDDEN_LINES** is **TRUE**, all the lines of every bar are visible.

DrawShaded

Surfaces: When DrawShaded is **TRUE**, Olectra Chart displays the data as a flat shaded surface in a 3D view with a Z-axis. The chart honors rotation and perspective control. The surface color is controlled with **XRT3D_SURFACE_BOTTOM_COLOR** and **XRT3D_SURFACE_TOP_COLOR**.

Bars: When DrawShaded is **TRUE**, Olectra Chart draws each bar as a solid bar. All bars with value greater than or equal to **XRT3D_ZORIGIN** will be drawn using **XRT3D_SURFACE_TOP_COLOR**, and all bars with value less than **XRT3D_ZORIGIN** will be drawn using **XRT3D_SURFACE_BOTTOM_COLOR**.

DrawContours

Surfaces: When DrawContours is **TRUE**, Olectra Chart examines the distribution of the data (using the **XRT3D_DISTN_METHOD** and possibly **XRT3D_DISTN_TABLE**), and draws contour lines demarcating each of the distribution levels. The contour line style, thickness and color are controlled with the **XRT3D_CONTOUR_STYLES** property.

Bars: When DrawContours is **TRUE**, Olectra Chart examines the distribution of the data (using **XRT3D_DISTN_METHOD** and possibly **XRT3D_DISTN_TABLE**), and draws contour lines around the bars, demarcating each of the distribution levels. The contour line style, thickness and color are controlled with the **XRT3D_CONTOUR_STYLES** property.

DrawZones

Surfaces: When DrawZones is **TRUE**, Olectra Chart examines the distribution of the data (using the **XRT3D_DISTN_METHOD** and possibly **XRT3D_DISTN_TABLE**), and fills each level with a solid color[1]. The color for each level is specified with the **XRT3D_CONTOUR_STYLES** property.

Bars: When DrawZones is **TRUE**, Olectra Chart examines the distribution of the data (using **XRT3D_DISTN_METHOD** and possibly **XRT3D_DISTN_TABLE**), and fills each level within each bar with a solid color.[1] The color for each level is specified with the **XRT3D_CONTOUR_STYLES** property. If **XRT3D_ZONE_DATA** is supplied, each bar is filled with a solid color. Otherwise, the bar is segmented by height according to the distribution table.

1. Unless DrawMesh is TRUE and DrawShaded is FALSE, in which case the fill color is used to draw each level's mesh lines.

The following table shows the 15 basic chart types:

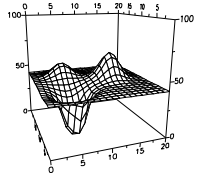
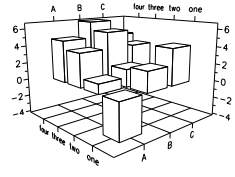
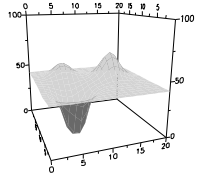
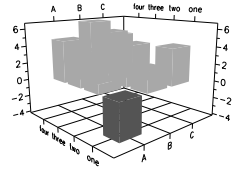
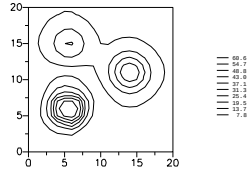
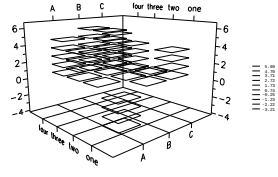
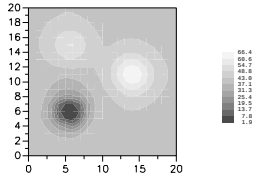
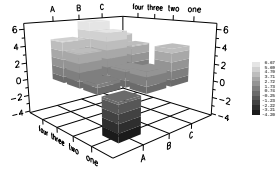
Chart Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface Example	Bar Example	Comments
1	T	F	F	F			**Mesh.** Displays surface as a mesh and bars in outline.
2	F	T	F	F			**Shaded.** Displays surface and bars in a flat shade. Top and bottom colors may be set.
3	F	F	T	F			**Contours.** Contour lines are automatically drawn between distribution levels in the data.
4	F	F	F	T			**Zones.**[a] Similar to #3, except that each distribution level is displayed in a solid color.

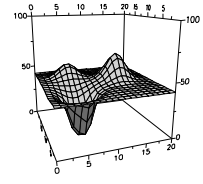
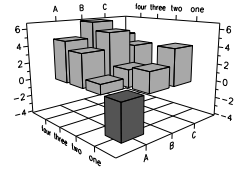
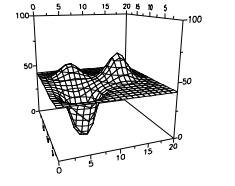
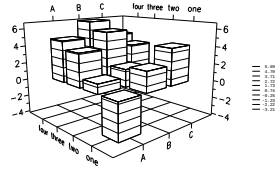
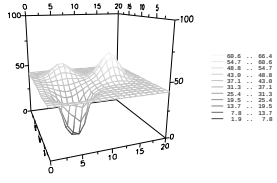
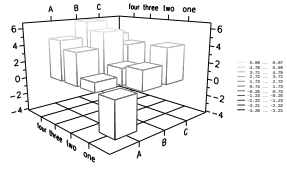
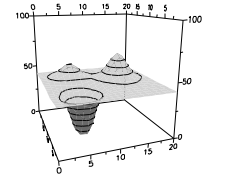
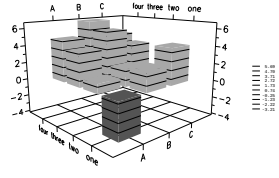
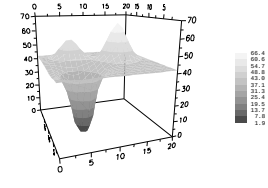
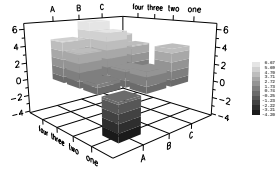
Chart Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface Example	Bar Example	Comments
5	T	T	F	F			**Mesh, Shaded.** Draws surface as a mesh and bars in outline. Surface and bars are flat shaded.
6	T	F	T	F			**Mesh, Contours.** Displays surface as a mesh and bars in outline. Also draws contour lines along borders between distribution levels in the data.
7	T	F	F	T			**Mesh, Zones.** Displays surface as a mesh and bars in outline Uses zoning colors for mesh and bar outlines.
8	F	T	T	F			**Shaded, Contours.** Displays a flat-shaded surface or bars with contour lines superimposed.
9	F	T	F	T			**Shaded, Zones.** Zone colors are used to flat shade the surface or bars.

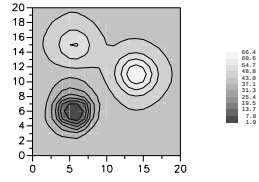
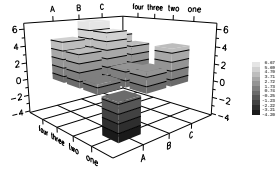
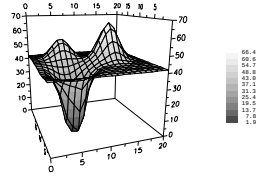
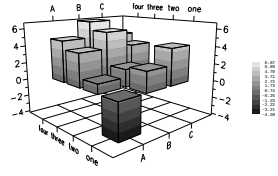
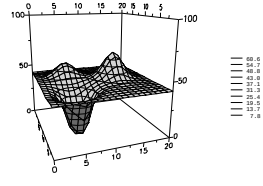
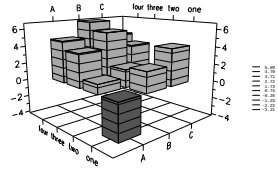
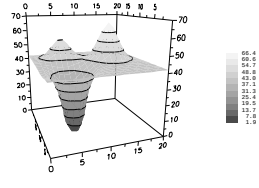
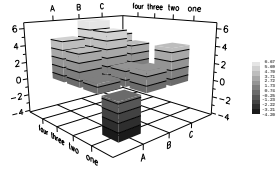
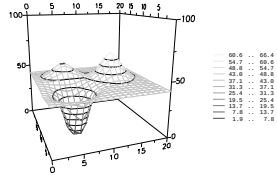
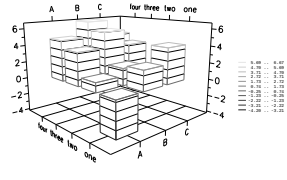
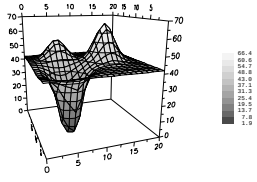
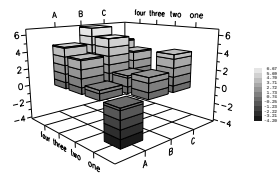
Chart Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface Example	Bar Example	Comments
10	F	F	T	T			**Contours, Zones.**[a] Displays contour lines and flat shaded zone colors to demarcate levels in the data.
11	T	T	F	T			**Mesh, Shaded, Zones.** Like #9, but with a mesh or bar outlines superimposed.
12	T	T	T	F			**Mesh, Shaded, Contours.** Like #5, but contour lines are superimposed.
13	F	T	T	T			**Shaded, Contours, Zones.** Like #9, but contour lines are superimposed.
14	T	F	T	T			**Mesh, Contours, Zones.** Like #7, but with contours superimposed.

Chart Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface Example	Bar Example	Comments
15	T	T	T	T			**Mesh, Shaded, Contours, Zones.** The sum of all basic options.

a. In this release, DrawZones is the same as DrawZones and DrawShaded for bar charts. Application developers are urged to use the DrawZones and DrawShaded combination for this view, since the interpretation of the DrawZones combination may change in a future release.

3.3 Bar Charts and Histograms

When **XRT3D_TYPE** is **XRT3D_TYPE_BAR**, the data pointed to by the **XRT3D_SURFACE_DATA** property will be displayed as a bar chart. Each data point will be represented by a single bar. Data for bar charts may be supplied in either a regular or irregular grid. In either case, the spacing between adjacent elements in the grid is honored.

Bar Z Origin

Bars start from the Z origin, which can be controlled through the **XRT3D_ZORIGIN** property. The default value of the origin is 0.0. When DrawShaded is **TRUE**, bars that have values greater than the origin are rendered in the **XRT3D_SURFACE_TOP_COLOR**. Negative bars (that is, bars with values less than the origin) are rendered in the **XRT3D_SURFACE_BOTTOM_COLOR**.

Bar Spacing

The amount of space occupied by a bar, as a percentage of the maximum amount possible, is controlled through the **XRT3D_[XY]BAR_SPACING** properties. The default is 80%. Setting it smaller results in thinner bars. Setting bar spacing to 100% (the maximum) results in bars that abut one another.

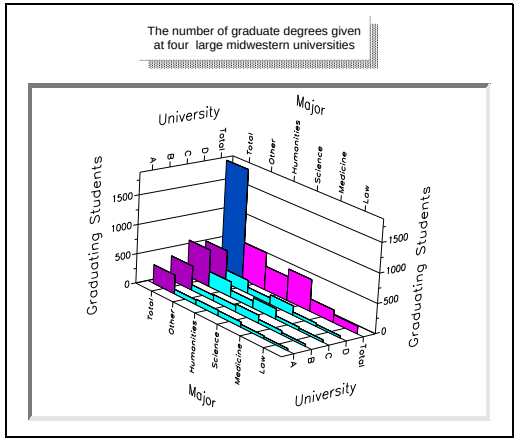


Figure 11 Fixed Bar Chart with X Spacing set to 1% and Y Spacing set to 100%

Histograms

To display a histogram, set **XRT3D_[XY]BAR_FORMAT** to **XRT3D_BAR_HISTOGRAM**. (The default is **XRT3D_BAR_FIXED**.) The X-axis and Y-axis can be independently switched between fixed and histogram formats. If the X-axis is switched to a histogram, each bar's left edge will be drawn aligned with corresponding X values in the data. The width of each bar will be the distance between subsequent X values in the data. Since the width of each bar is derived from the spacing between its neighbors, there is always one fewer bar along a histogram axis than there is if the same data is displayed along a fixed axis.

Histograms usually make use of irregularly-gridded data. This gives control over spacing in the data grid. See section 4.2 on page 49 for more information on irregularly-gridded data.

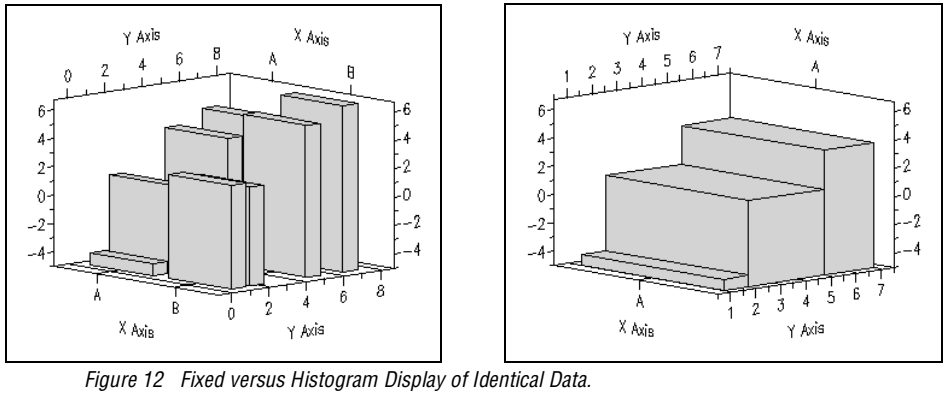


Figure 12 Fixed versus Histogram Display of Identical Data.

Bar Colors

If DrawShaded is **TRUE** and DrawZones is **FALSE**, the colors of the individual bars can be controlled. Normally, all bars with value below the origin are colored with the surface bottom color, and all others are colored with the surface top color. However, in some situations, it is useful to color a row, column or individual bar in the chart with a distinct color.

The property **XRT3D_XY_COLORS** is used to specify the fill color of a line of bars, or of an individual bar. It takes a pointer to a **NULL**-terminated list of pointers to the following structure:

```
typedef struct {
    int             xindex;     /* 0-indexed */
    int             yindex;     /* 0-indexed */
    COLORREF        *color;     /* color  */
} Xrt3dXYColor;
```

An individual bar can be colored by specifying its indices in the structure. For example, to set the bar in the third X data line, second point to red, specify the values (**2, 1, RGB**(*255*,**0,0**)) in one of the **Xrt3dXYColor** structures. To specify the color of an entire row of bars, set the other index to -1. For instance, the entire fifth X line of bars is set to green with (**4, -1, RGB**(**128**,*255*,**128**)). To set the color of all the bars, set both *xindex* and *yindex* to -1.

You can retrieve the color used for a bar by calling the **Xrt3dGetXYColor**() procedure. The **Xrt3dSetXYColor**() procedure makes it easy to change the bar coloring. For example, to change the fourth row of bars (as measured along the Y-axis) to yellow, you could use:

```
Xrt3dSetXYColor(mygraph, -1, 3, RGB(255, 255, 0));
```

Olectra Chart will only maintain one entry per (*xindex*, *yindex*) combination. Whenever a second entry for the same indices is supplied, the first entry is removed. Later entries take precedence over earlier entries.

If any **Xrt3dXYColor** structure contains a **NULL** for the *color*, it is taken as a request to delete any other entry with the same indices. This is most conveniently used with the procedure **Xrt3dSetXYColors**(), to remove individual color entries.

XYColors are not placed in the legend. This method of coloring bars should only be used if it is not important to label the bars a certain color. If labelling is important, a 4D bar chart should be created. 4D bar charts are discussed in section 6.2 on page 66.

3.4 Contours and Zone Display

When DrawContours or DrawZones is **TRUE**, Olectra Chart marks each distribution level from an array of 100 built-in contour styles.

Each contour style contains information about the contour line style, width, pattern, color, and zone color (used to mark each *level*). Contour styles can be customized by an application. For more information, see section 6.6 on page 73.

Contour Styles Used

Olectra Chart determines which contour style to use for a particular level automatically, evenly distributing the styles through the number of levels, as shown by Figure 13.

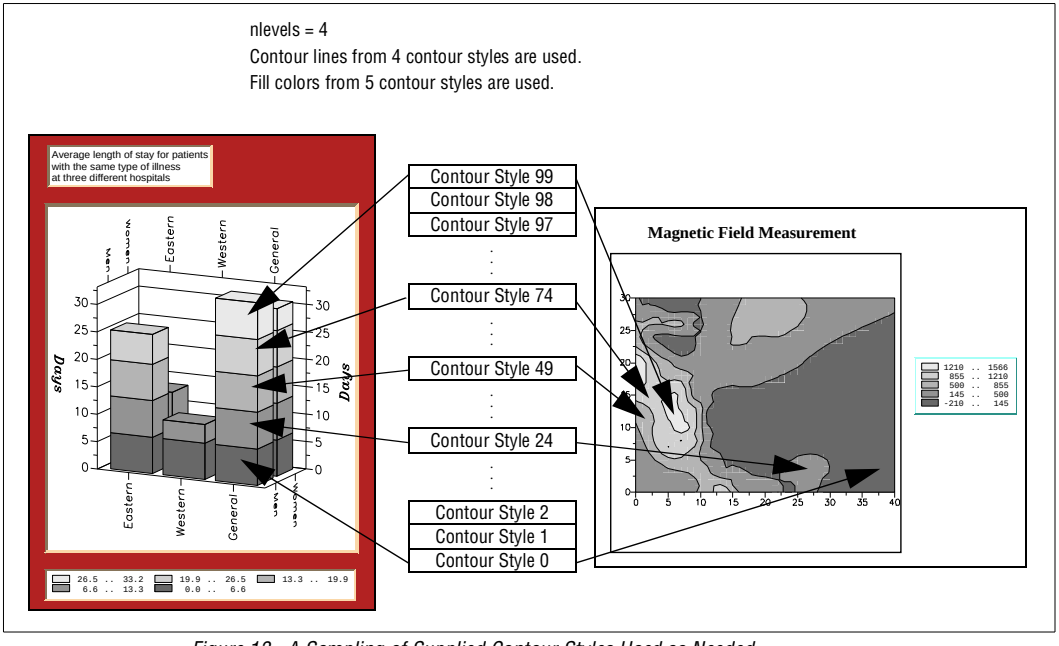


Figure 13 A Sampling of Supplied Contour Styles Used as Needed

Contour/Zone Projection

The contours and zones determined for the chart can be displayed on the top or bottom side of the unit cube using the **XRT3D_PROJECT_ZMAX** and **XRT3D_PROJECT_ZMIN** properties. These properties take integers combined from the following constants:

```
#define XRT3D_PROJECT_CONTOURS  0x1
#define XRT3D_PROJECT_ZONES     0x2
```

For instance, to project contours and zones onto the plane *z=zmax*, set **XRT3D_PROJECT_ZMAX** to the value (**XRT3D_PROJECT_CONTOURS|XRT3D_PROJECT_ZONES**). To remove all projections from a side, set the property to 0.

These properties are not dependent on the values of DrawContours or DrawZones. However, any property that affects contour generation or rendering (such as **XRT3D_CONTOUR_STYLES**, **XRT3D_NUM_DISTN_LEVELS** or **XRT3D_ZONE_DATA**) affects projected contours/zones. **XRT3D_PROJECT_ZMAX** and **XRT3D_PROJECT_ZMIN** are ignored in 2D charts and bar charts.

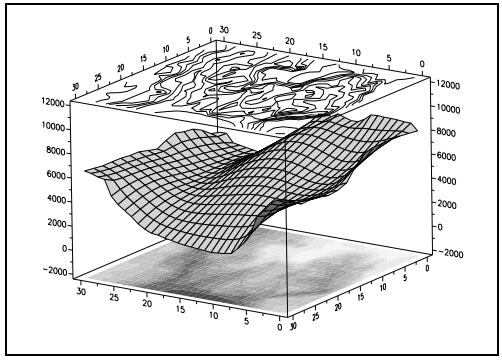


Figure 14 Projecting Contours and Zones

Zoning Method

An application can control the method used to fill each zone region with the **XRT3D_ZONE_METHOD** property. By default (**XRT3D_ZONE_CONTOURS**), Olectra Chart fills between each contour interval. When set to **XRT3D_ZONE_CELLS**, Olectra Chart fills entire cells in the grid based on the average of the four corners of the cell. Figure 15 illustrates the visual difference.

Cell zoning produces a coarser-looking surface, but offers significant performance advantages over contour zoning. However, the visual difference between the two methods diminishes with larger grids.

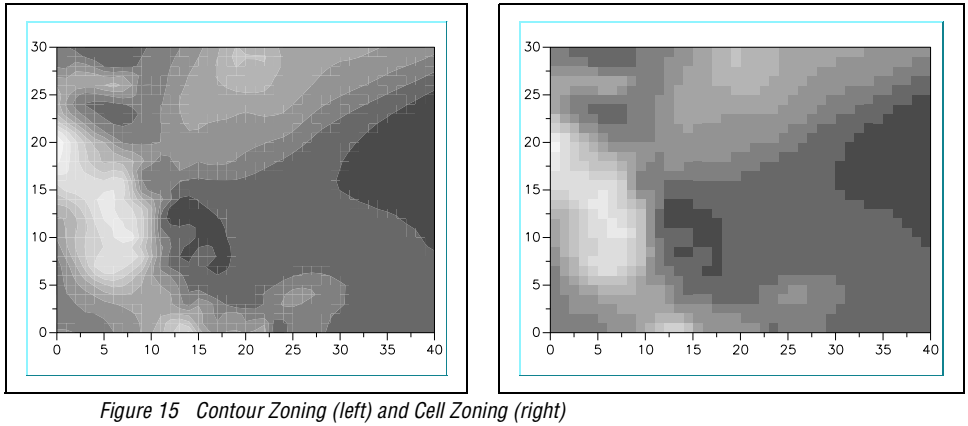


Figure 15 Contour Zoning (left) and Cell Zoning (right)

3.5 Axis Controls

The properties listed in Figure 5 on page 21 control the Axis display.

Axis Show

The **XRT3D_[XYZ]AXIS_SHOW** properties tell Olectra Chart whether it should draw the axis at all. If set to **FALSE**, the axis will not be drawn.

Axis Font and Size

The axis annotation is rendered using the stroke font specified by **XRT3D_AXIS_STROKE_FONT**. There are several stroke fonts to choose from (see Appendix E). The default font is **XRT3D_SF_ROMAN_SIMPLEX**.

The axis font size is measured in units which are each 1/1,000 of the unit cube length. The default axis font size is 80, which means the characters are 8% of the length of the unit cube high. See section 2.1 on page 11 for a definition of the unit cube.

Title

The **XRT3D_[XYZ]TITLE** properties may be used to specify a title for each axis. Titles are rendered in the stroke font specified by **XRT3D_[XYZ]AXIS_TITLE_STROKE_FONT**, and in the size specified by **XRT3D_[XYZ]AXIS_TITLE_STROKE_SIZE**.

Min and Max

The **XRT3D_[XYZ]MAX** and **XRT3D_[XYZ]MIN** properties specify the minimum and maximum values of each axis. By default, Olectra Chart determines the extent of the axes based on the minimum and maximum data values and anytime the data changes, the axes update automatically. You can frame the data displayed by setting these properties. When set by an application, the axis extents do not change when the data changes. To return to the default behavior, set **XRT3D_[XYZ]MAX_USE_DEFAULT** or **XRT3D_[XYZ]MIN_USE_DEFAULT** to **TRUE**.

Note: The Z-axis minimum/maximum cannot be inside the Z-range of the data. When **XRT3D_ZMIN**/**XRT3D_ZMAX** has been set, any changes to the data that put it inside the Z-range causes Olectra Chart to set its corresponding **USE_DEFAULT** property to **TRUE**.

3.6 Legend Display

Olectra Chart will automatically generate a legend whenever contours or zones are drawn. If you don't want a legend, set **XRT3D_LEGEND_SHOW** to **FALSE**. If contours are drawn and zones are not, a legend listing the contour lines is generated. If zones are drawn, the legend lists the fill colors for each level.

By default, Olectra Chart will attempt to list the legend contents vertically and position the legend to the right (i.e. east) of the graph area.

Orientation

The legend layout is controlled through the **XRT3D_LEGEND_ORIENTATION** property. It may be either **XRT3D_ALIGN_VERTICAL** or

XRT3D_ALIGN_HORIZONTAL. If the legend is too large to fit in one row or column, Olectra Chart will create the legend in several rows or columns.

Anchor

The default legend positioning relative to the graph area is controlled with the **XRT3D_LEGEND_ANCHOR** property. Valid values correspond to the eight points of the compass: **XRT3D_ANCHOR_NORTH**, **XRT3D_ANCHOR_SOUTH**, **XRT3D_ANCHOR_EAST**, **XRT3D_ANCHOR_WEST**, **XRT3D_ANCHOR_NORTHWEST**, **XRT3D_ANCHOR_NORTHEAST**, **XRT3D_ANCHOR_SOUTHEAST**, and **XRT3D_ANCHOR_SOUTHWEST**.

Style

The **XRT3D_LEGEND_STYLE** property can be set to either **XRT3D_LEGEND_STYLE_CONTINUOUS** (the default) or **XRT3D_LEGEND_STYLE_STEPPED**. See Figure 16 for an example of these styles. Continuous legends are recommended since they display the same information while using less space.

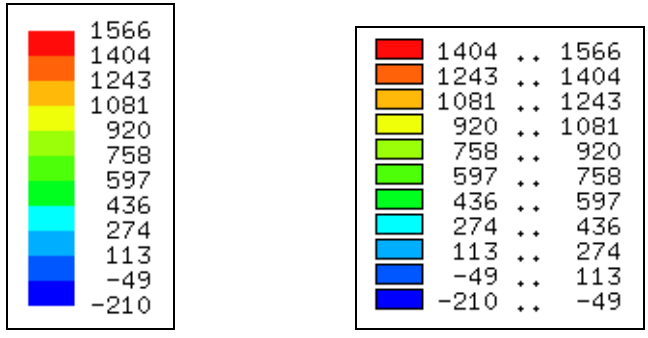


Figure 16 Continuous (left) and Stepped (right) Legends

Legend Position and Border

To explicitly position the legend, see section 3.16 on page 41. Also, see section 3.17 on page 42 for information on how to program the legend border.

Custom Legend Contents

It is possible to specify custom text for the legend elements. See section 6.5 on page 71 for more information.

3.7 Perspective

The **XRT3D_PERSPECTIVE_DEPTH** property controls the perspective effect observed by projecting the unit cube onto the screen. Small values exaggerate the perspective effect, while large values diminish it. Valid values are anything between 1 and **XRT3D_HUGE_VAL**.

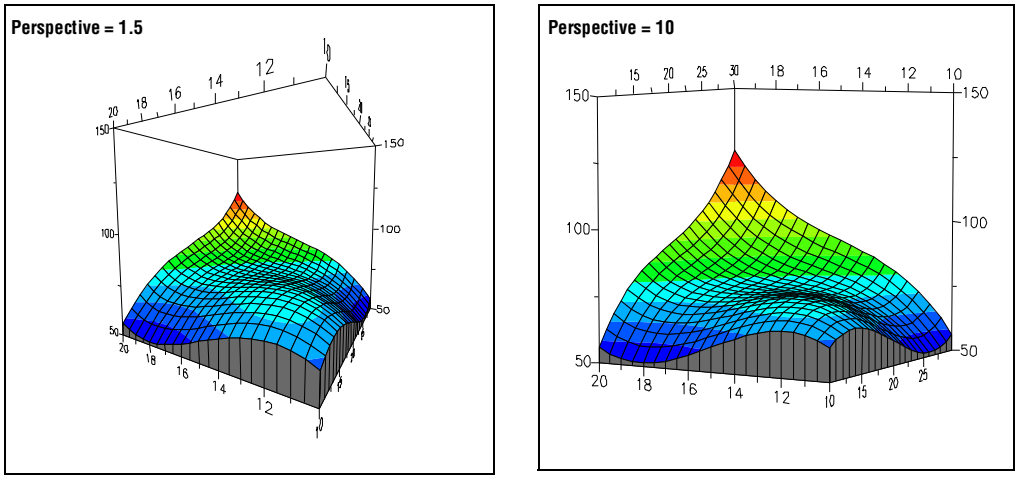


Figure 17 Perspective Depth Measurement

3.8 Mesh Controls

Mesh Colors

The bottom and top colors of the mesh drawn when DrawMesh is **TRUE** can be set with **XRT3D_MESH_BOTTOM_COLOR** and **XRT3D_MESH_TOP_COLOR**. They are both "Black" by default.

Mesh Filtering

The **XRT3D_[XY]MESH_FILTER** properties specify how the mesh is filtered before being displayed. By default, no filtering is performed. When set to 0, Olectra Chart automatically filters the mesh to provide a pleasing display, and changes the filter as the chart is scaled or the data changes.

You can hard-code a mesh filter by setting these properties to any positive integer. For example, a value of 5 filters the mesh so that every 5th line is drawn.

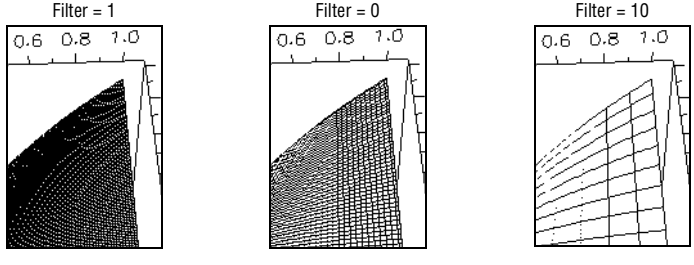


Figure 18 Effect of Mesh Filtering

Hidden Mesh Lines

When DrawMesh is **TRUE** and DrawShaded is **FALSE**, grid and contour lines that are obscured from view by intervening portions of the scene are not displayed by default. To display these lines, set **XRT3D_DRAW_HIDDEN_LINES** to **TRUE**.

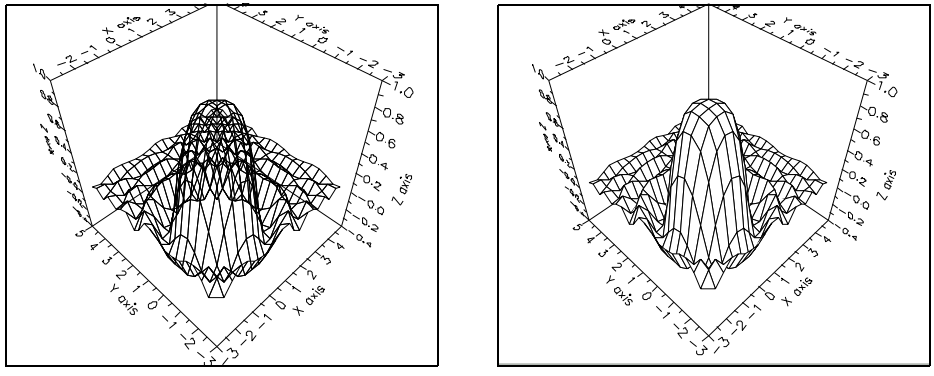


Figure 19 Hidden Line Removal

3.9 Surface Colors

The bottom and top colors of the shaded surface drawn when DrawShaded is **TRUE** can be set with **XRT3D_SURFACE_BOTTOM_COLOR** and **XRT3D_SURFACE_TOP_COLOR**. By default the bottom color is RGB(112,112,112) and the top color is RGB(211,211,211).

3.10 Solid Surface

Setting **XRT3D_SOLID_SURFACE** to **TRUE** will cause Olectra Chart to draw a skirt around the data, thereby joining the edge of the surface to a plane at the minimum Z value, as shown in Figure 20.

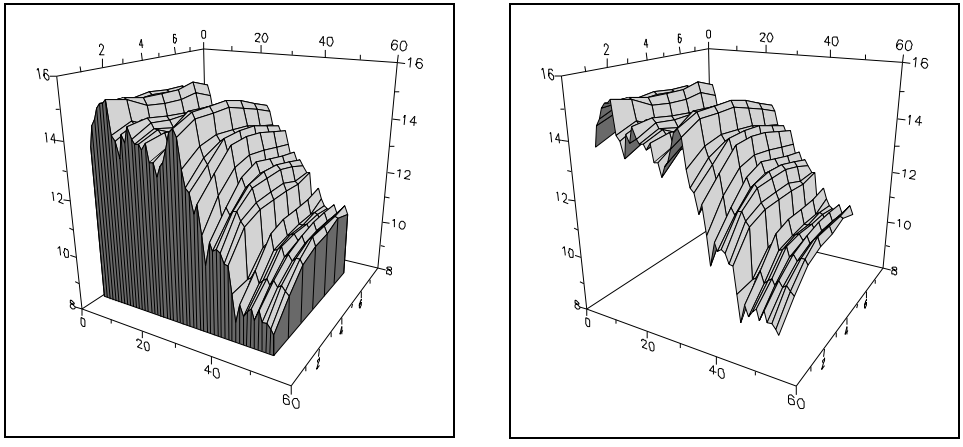


Figure 20 Setting Solid Surface On and Off

3.11 Axis Scaling

The axis scaling properties **XRT3D_[XYZ]SCALE** can be used to adjust the X, Y and Z dimensions of the unit cube relative to one another. For example, if you would like the 3D display to be twice as long in the Y direction as in the X, set the Y scale to twice the X scale. Setting the Z scale higher or lower has the effect of flattening or stretching the surface view.

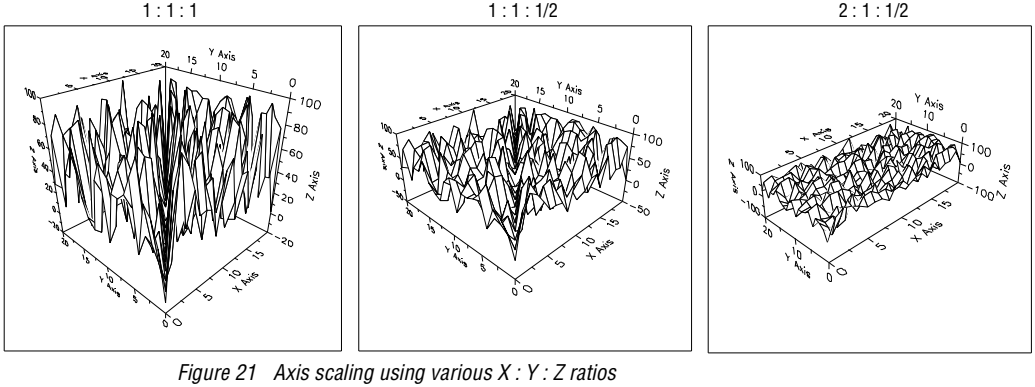


Figure 21 Axis scaling using various X : Y : Z ratios

3.12 Axis Labelling

There are three distinct ways to annotate the X- and Y-axes, and two ways the Z-axis can be annotated. The annotation method is determined by the

XRT3D_[XYZ]ANNO_METHOD property which takes one of the following three values:

```
typedef enum {
    XRT3D_ANNO_VALUE,
    XRT3D_ANNO_DATA_LABELS,     /* X, Y axes only */
    XRT3D_ANNO_VALUE_LABELS
} Xrt3dAnnoMethod;
```

Value Method

When **XRT3D_[XYZ]ANNO_METHOD** is **XRT3D_ANNO_VALUE**, Olectra Chart will automatically annotate the axis based on the range of data. It is most suitable for the Z-axis, and for the X- and Y-axes when the value of **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**. This is the default annotation method.

Data Labels Method

To label individual lines in a surface, or a row/column of bars, use **XRT3D_ANNO_DATA_LABELS**. This method uses a list of strings supplied with the **XRT3D_[XY]DATA_LABELS** property to annotate each line from the grid. Individual labels can be supplied with the procedure **Xrt3dSetNthDataLabel**() and retrieved with **Xrt3dGetNthDataLabel**() procedure.

Value Labels Method

Labels can be placed at explicit locations along an axis by setting the value of **XRT3D_[XYZ]ANNO_METHOD** to **XRT3D_ANNO_VALUE_LABELS**. The locations of the labels are specified with the property **XRT3D_[XYZ]VALUE_LABELS**, which takes an array of pointers to the following structure:

```
typedef struct {
    double      value;
    char *      label;
} Xrt3dValueLabel;
```

Any labels which are beyond the bounds of the axis are not drawn. Individual value labels can be added and removed through the procedure **Xrt3dSetValueLabel**(), and retrieved through the procedure **Xrt3dGetValueLabel**().

3.13 Grid Lines

Grid lines can be displayed on each of the three primary planes using the **XRT3D_[XYZ]GRID_LINES** properties. These properties take integers as values, combined from the following constants:

```
#define XRT3D_XY_PLANE      1
#define XRT3D_XZ_PLANE      2
#define XRT3D_YZ_PLANE      4
```

For instance, to draw X grid lines in all applicable planes, set **XRT3D_XGRID_LINES** to the value (**XRT3D_XY_PLANE | XRT3D_XZ_PLANE**). To remove grid lines completely, set the value to 0. In a 2D chart, grid lines are displayed whenever the **XRT3D_[XY]GRID_LINES** property is non-zero.

Grid lines are drawn where annotation is drawn on the axis, regardless of the annotation method. Grid lines are always drawn as a solid 1-pixel wide line, using **XRT3D_GRAPH_FOREGROUND_COLOR** or, if that is **NULL**, **XRT3D_FOREGROUND_COLOR**.[1]

3.14 Header and Footer Text

The header and footer areas can both contain multiple lines of text. Text is aligned within the area, depending on the value of the **XRT3D_HEADER_ADJUST** or **XRT3D_FOOTER_ADJUST** property. The values **XRT3D_ADJUST_LEFT**, **XRT3D_ADJUST_RIGHT** and **XRT3D_ADJUST_CENTER** cause the text to be left-justified, right-justified, or centered. **XRT3D_ADJUST_CENTER** is the default.

Text for the header and footer areas is specified using the **XRT3D_HEADER_STRINGS** and **XRT3D_FOOTER_STRINGS** properties. Both of these properties have **NULL**-terminated arrays of strings as their values.

The code below will set two header lines and left-align them:

```
static char *hs[] = { "Analysis Results",
                      "Sample Group A", NULL };
Xrt3dSetValues(my_graph,
    XRT3D_HEADER_STRINGS,   hs,
    XRT3D_HEADER_ADJUST,    XRT3D_ADJUST_LEFT,
    NULL);
```

3.15 Header, Footer and Legend Fonts

A font may be specified for each of the header, footer and legend areas. Olectra Chart can use any font available on the system at runtime.

Use the **CreateFont**() or **CreateFontIndirect**() Windows API call to create an **HFONT** structure for use with Olectra Chart's font properties. The Windows API **EnumFontFamilies**() function determines which fonts are available on the system. Consult your Windows programming documentation for further details on finding and setting fonts.

Another way to set a font property is to use the **Xrt3dSetPropString**() function. This allows you to avoid creating and destroying an **HFONT** or **LOGFONT** structure and set a font using a simple string, such as "Arial,24,Italic".

1. A future release of Olectra Chart may support customizable line styles and colors for the grid lines.

The following example uses both methods to set font properties:

```
/* Use CreateFont() to set Header font */
HFONT   hFont;
hFont = CreateFont(24, 0, 0, 0, 0,  /* Set only Size & */
    0, 0, 0, 0, 0, 0, 0, 0, "MS Serif");    /* Typeface */
Xrt3dSetValues(hChart, XRT3D_HEADER_FONT, hFont, NULL);
DeleteObject(hFont);

/* Use Xrt3dSetPropString() to set Legend font */
Xrt3dSetPropString(hChart, XRT3D_LEGEND_FONT,
            "Times,12,bold");
```

For information on setting font properties in the Windows 3.1 environment, refer to section 2.6 on page 15.

3.16 Area Positioning

At run-time, each of the four areas (header, footer, legend and graph) will, by default, be positioned by Olectra Chart. The default positioning for each area depends on a large number of factors, including:

- The control's current width and height.
- The size of the legend, header and footer areas. These, in turn, depend on the text and fonts being used.
- The value of the **XRT3D_LEGEND_ANCHOR** property.
- The positioning of areas which have been explicitly positioned by the user program.

Olectra Chart's default positioning algorithms will size and position the header, footer and legend areas first. The graph area will be sized and positioned to fit into the largest remaining rectangular area.

A program can determine and adjust area positioning through the use of the positioning properties.

XRT3D_HEADER_X	XRT3D_HEADER_Y	(XRT3D_HEADER_WIDTH)	(XRT3D_HEADER_HEIGHT)
XRT3D_FOOTER_X	XRT3D_FOOTER_Y	(XRT3D_FOOTER_WIDTH)	(XRT3D_FOOTER_HEIGHT)
XRT3D_LEGEND_X	XRT3D_LEGEND_Y	(XRT3D_LEGEND_WIDTH)	(XRT3D_LEGEND_HEIGHT)
XRT3D_GRAPH_X	XRT3D_GRAPH_Y	XRT3D_GRAPH_WIDTH	XRT3D_GRAPH_HEIGHT

(Properties in Parentheses are Read-Only)

Figure 22 Area Positioning Properties

When used in an **Xrt3dGetValues**() call, positioning properties will return the values used the last time the control was displayed. The width and height properties are of type **int**, and the X and Y properties are of type **int**.

A program should not explicitly set any of the positioning properties unless it is prepared to recalculate them when the control's size changes. See section 5.8 on page 63 for more information on handling window resizing.

In some situations, it may be worthwhile to explicitly set some or all of the positioning properties. For example, if the program will be displaying data that changes in real time, the default positioning may change slightly with each redisplay. These small positioning changes could distract the user. In this situation, the program should use one of the following strategies:

- Hardcode all the positioning properties. Window resizing should not be allowed; or
- Let Olectra Chart calculate default positioning for all areas when the control first displays, and for the first display after any window resize. After the first display, explicitly set the positioning properties to the values calculated by Olectra Chart.

3.17 Area Borders

Each of the 4 graph areas (header, footer, legend and graph) may be enhanced with a border. There is a Border and a BorderWidth property for each of these areas (for example, **XRT3D_GRAPH_BORDER** and **XRT3D_GRAPH_BORDER_WIDTH**). In addition, you can specify a border for the entire control.

The Border properties may be set to any of **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW** and **XRT3D_BORDER_PLAIN**. The width of the border (in pixels) is controlled with the corresponding BorderWidth property. The width must be between 0 and 20.

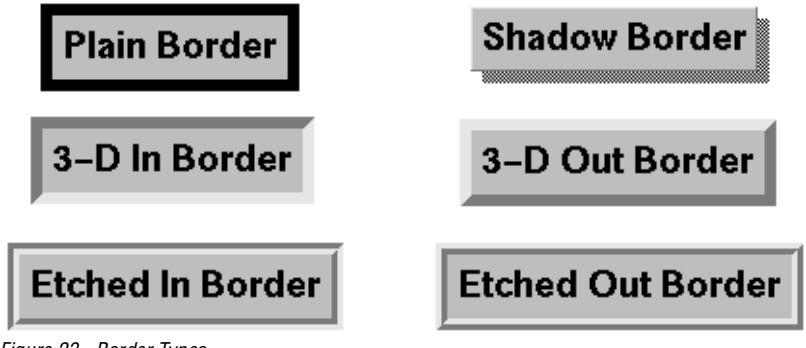


Figure 23 Border Types

If control areas are explicitly positioned to intersect each other, highlighting the intersections by showing borders may look unattractive.

3.18 Foreground and Background Colors

Olectra Chart supports the specification of colors for the window background and foreground, as well as for the lines and bars that represent data in the chart itself. Olectra Chart will choose default colors for the application, so simple applications need not concern themselves with color specification.

Background Colors

The window background color is specified through the **XRT3D_BACKGROUND_COLOR** property. It is white by default.

Each of the header, footer, legend and graph areas also have a background color which is **XRT3D_DEFAULT_COLOR** (transparent) by default. For example, the property to change the legend background color is **XRT3D_LEGEND_BACKGROUND_COLOR**. The data area of the chart also has its own background color, specified by **XRT3D_DATA_AREA_BACKGROUND_COLOR**.

Foreground Colors

The window foreground color is black by default, and is specified with the **XRT3D_FOREGROUND_COLOR** property. It will be used as the foreground color for each of the header, footer, legend and graph areas, unless a different foreground color is specified for one or more of these areas. For example, to specify a different header foreground color, use the **XRT3D_HEADER_FOREGROUND_COLOR** property.

Specifying Colors

All color properties take a valid Windows color reference as their value. Use **Xrt3dSetValues**() to set a property to a color reference created with the **RGB** macro.

Alternately, use **Xrt3dSetPropString**() to set a property to a named color string, as shown by the following example:

```
Xrt3dSetPropString(hChart,
    XRT3D_BACKGROUND_COLOR,"skyblue");
```

Olectra Chart recognizes over 200 colors (ranging from “Aquamarine” to “YellowGreen”). A list of recognized colors can be found in the file OC_COLOR.H, located in Olectra Chart’s \INCLUDE directory.

Palette Handling

Because the charts created by Olectra Chart look best when rendered using solid colors, Olectra Chart automatically adds new solid colors to the Windows palette when creating the chart or changing color properties. This saves the programmer the step of allocating a new color in the palette before setting a color property. If the palette is full, the color is set to the nearest palette color or by dithering the closest palette colors, depending on the macro or function you used to specify the color.

Once a chart is created, an application should not update or change the Windows palette directly. Changes to any of the chart’s colors should be made by updating the appropriate chart properties (such as **XRT3D_BACKGROUND_COLOR**).

The Windows color palette is a shared resource. In some situations, the number of colors in use by all the applications being displayed is more than the number of palette entries available. In this case, the colors in some windows will "flash" to inappropriate colors as the user uses different applications.

Palette Notification Message

To ensure proper color palette handling, your application needs to handle the **XRT3DN_PALETTECHANGED** notification message, as well as **WM_QUERYNEWPALETTE** and **WM_PALETTECHANGED**. The **XRT3DN_PALETTECHANGED** message is sent to a chart's parent window after the chart control has changed its color palette. For more information on this message, refer to Appendix D. The SIMPLE.C example in Appendix F provides an example of a program which handles **XRT3DN_PALETTECHANGED**.

3.19 Double Buffering

Double buffering is a graphics technique which will reduce the amount of flashing perceived by a user when a chart changes.

When **XRT3D_DOUBLE_BUFFER** is **TRUE**, every time Olectra Chart changes a chart, it will:

- allocate (if necessary) and clear an off-screen bitmap.
- render the complete image to the off-screen bitmap.
- copy the off-screen bitmap to the screen.

When **XRT3D_DOUBLE_BUFFER** is **FALSE**, every time Olectra Chart changes a chart it will clear the screen image (possibly causing a visual flash) and then render the complete image to the visible window (possibly allowing the user to see the chart being drawn piece by piece).

By default, **XRT3D_DOUBLE_BUFFER** is **TRUE**. However, setting it to **FALSE** can improve the graphing performance and reduce memory requirements.

3.20 Output and Printing

Many applications need to provide users with a way to get a hardcopy of a chart they see on the screen. Olectra Chart provides several procedures that output chart representations to files or printers. These procedures work correctly even when the chart control is obscured by other windows. These procedures are fully described in Appendix C.

Output/Printing Procedures

The procedures listed below are fully documented in Appendix B on page 95:

Xrt3dDrawToClipboard()	Outputs a chart image to the Windows clipboard using the graphics format you specify.
Xrt3dDrawToDC()	Outputs a chart image to any device context (DC) at the scale and graphics format you specify.
Xrt3dDrawToFile()	Outputs a chart image to a file using the graphics format you specify.
Xrt3dPrint()	Outputs a chart image to a printer, using the Windows Print dialog box.

Printing Charts

To print a chart using the standard Windows Print dialog box, call **Xrt3dPrint**(). This procedure allows you to specify the image format and the size/position as described above; its only difference is that it displays the Print dialog box.

Xrt3dDrawToDC()

Use the **Xrt3dDrawToDC**() procedure for more complex chart output. For instance, to print several charts on one page, call **Xrt3dDrawToDC**() for each chart, using the *top*, *left*, *width* and *height* arguments to specify each chart's different size/position. Figure 24 shows using **Xrt3dDrawToDC**() for a printing function that prints a chart to the default printer without using the Windows Print dialog box.

```
BOOL
printGraph(HXRT3D hChart)
{
    PRINTDLGpd;
    DOCINFO di;
    HDC     hdc;
    TEXTMETRIC tm;
    RECT    rect;

    memset ((void *) &pd, 0, sizeof(pd));
    pd.lStructSize = sizeof(pd);
    pd.hwndOwner   = NULL;
    pd.Flags       = PD_RETURNDC | PD_RETURNDEFAULT;
    pd.hInstance   = NULL;
    PrintDlg(&pd);
    hdc = pd.hDC;

    if (!hdc) {
        return (FALSE);
    }

    di.cbSize = sizeof(di);
    di.lpszDocName = "My Graph";
    di.lpszOutput = NULL;

    StartDoc(hdc, &di);
    StartPage(hdc);

    GetTextMetrics(hdc, &tm);

    /* Define graph size and output graph */
    rect.left = tm.tmAveCharWidth * 2;
    rect.top = (tm.tmHeight + tm.tmExternalLeading) * (3);
    rect.right = tm.tmAveCharWidth * 42;
    rect.bottom = (tm.tmHeight + tm.tmExternalLeading) * (25);

    Xrt3dDrawToDC(hChart, hdc, XRT3D_DRAW_METAFILE,
        XRT3D_DRAWSCALE_NONE, rect.left, rect.top,
        rect.right-rect.left, rect.bottom-rect.top);

    EndPage(hdc);
    EndDoc(hdc);
    DeleteDC(hdc);
    return (TRUE);
}
```

Figure 24 Non-interactive chart printing procedure

4

Olectra Chart Data

Data Overview ■ *The Xrt3dData Structure*

4.1 Data Overview

Data to be graphed can originate from diverse sources: files, databases, real time data feeds, or even unrelated processes running on the machine. The data can represent the results of an experiment or the solution to an analytical problem. Olectra Chart displays data graphically. It does not do any analysis of the data, except for possibly displaying its zone and contour distributions.

This chapter discusses the **Xrt3dData** structure in detail and offers examples of allocating and loading the **Xrt3dData** structure.

Rules and Guidelines

Olectra Chart graphs surfaces that are increasing in X and Y. It does not graph surfaces that fold back in X or Y (such as a sphere). Olectra Chart expects its data to come from a regularly or irregularly gridded surface. Think of X and Y gridlines forming a mesh. When gridlines meet, they define a point which has a Z value. If a point doesn't have a Z value, it is called a *hole*.

Olectra Chart is optimized for small dimension meshes. Ideally, grids should not be more than 100 x 100. Very large dimension data will take longer to graph, and the results may not be satisfactory. If you need to graph large dimension data, thin it out before using it with Olectra Chart. For example, you might decide to load every tenth grid point into the **Xrt3dData** structure or use the **Xrt3dDataWindow**() procedure.

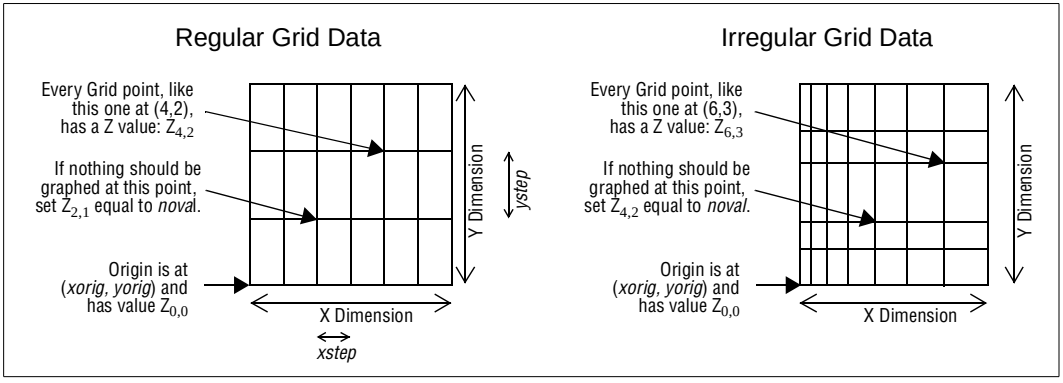


Figure 25 7 x 4 Regular grid and 8 x 6 Irregular grid

Holes in the Data

The **Xrt3dData** structure contains an element called *noval*. Whenever Olectra Chart sees values in the grid data that are the same as *noval*, it will treat that grid intersection as a hole. You should set *noval* to be a value that is well outside of the range of your data. For example, you may want to set it to **XRT3D_HUGE_VAL**.

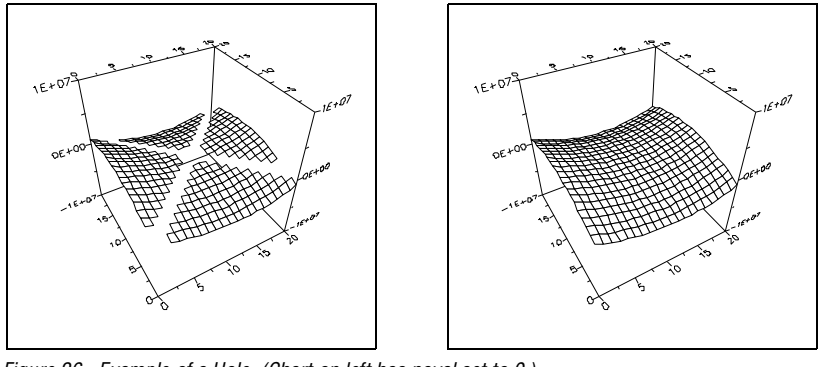


Figure 26 Example of a Hole. (Chart on left has noval set to 0.)

Data from a File

If the data exists in a file, and it does not need to be changed or updated in real time, there are two options to consider. The first option is to massage the data so that the file conforms to a syntax understood by **Xrt3dMakeDataFromFile**(). This procedure will allocate the **Xrt3dData** structure and load it with data from a named file. **Xrt3dMakeDataFromFile**() is fully described in Appendix C. There is also a corresponding **Xrt3dSaveDataToFile**() procedure.

Another approach is to allocate an **Xrt3dData** structure using **Xrt3dMakeGridData**() or **Xrt3dMakeIrGridData**() and populate it with data by reading the data file (perhaps by using **fgets**() or **fscanf**()).

Changing Data

It is easy to change data while leaving other control properties unchanged. Programs can create any number of **Xrt3dData** structures and then switch among them by setting the **XRT3D_SURFACE_DATA** property to point to the current data.

Real Time Performance

For the best possible real time performance, try to keep the grid dimensions small. DrawShaded and DrawZones are the slowest types of displays. DrawMesh is the fastest type of display. If **XRT3D_DRAW_HIDDEN_LINES** is **TRUE**, Olectra Chart uses an optimized algorithm to speed display even further. Keeping the window size small and turning off double buffering will also improve performance. Use of **XRT3D_REPAINT** can also improve performance dramatically.

Responsibility for Data

It is the application's responsibility to allocate and destroy all required **Xrt3dData** structures. Olectra Chart does not make a copy of the data; instead, it references data in the application's memory pointed to by **XRT3D_SURFACE_DATA** and by **XRT3D_ZONE_DATA**.

If Olectra Chart has to repaint the control window, it will use the data pointed to by **XRT3D_SURFACE_DATA**. An application may adjust the values inside an **Xrt3dData** structure while that structure is in use, if it can ensure that the structure will be correct when next required by Olectra Chart. Remember to reset the **XRT3D_SURFACE_DATA** property after changing any data values so that Olectra Chart refreshes the display.

4.2 The Xrt3dData Structure

Whenever Olectra Chart displays a chart image, it uses the data values located in the **Xrt3dData** structure pointed to by the **XRT3D_SURFACE_DATA** property.

The **Xrt3dData** structure is defined in the OLCH3DCM.H header file. (It is also defined in Appendix E.) The **Xrt3dData** structure is defined as a union of regular and irregular grid structures:

```
typedef union{
    Xrt3dGridData   g; /* regular grid */
    Xrt3dIrGridData ig; /* irregular grid */
} Xrt3dData;
```

Regular Grid Data

The **Xrt3dGridData** structure is defined as follows:

```
typedef struct {
    Xrt3dDataType   type; /*XRT3D_DATA_GRID*/
    int             numx, numy;
    double          noval;
    double          xstep, ystep;
    double          xorig, yorig;
    double          **values;
}Xrt3dGridData;
```

The **Xrt3dGridData** structure contains a **type** field which is always initialized to **XRT3D_DATA_GRID** by **Xrt3dMakeGridData**() and **Xrt3dMakeDataFromFile**().

numx and *numy* are integers defining the number of grid lines (i.e. elements) in the X and Y dimensions. For surface and histogram charts, *numx* and *numy* must be ≥ 2. For bar charts, *numx* and *numy* must be ≥ 1.

The special value which Olectra Chart treats as "no value" is specified by *noval*. The distance between grid lines in the X and Y direction is specified by *xstep* and *ystep*, and should always be > 0. The origin of the entire grid is specified with *xorig* and *yorig*.

values is a two-dimensional array of the Z values for each grid point in X major order, as Figure 27 illustrates.

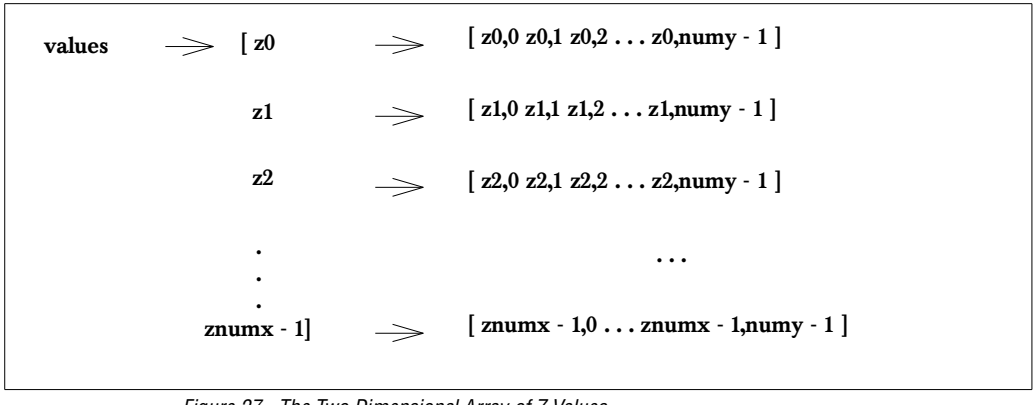


Figure 27 The Two Dimensional Array of Z Values

The easiest way to allocate the **Xrt3dData** structure is by using **Xrt3dMakeGridData**(). This procedure is described in Appendix C.

Irregular Grid Data

The **Xrt3dIrGridData** structure is defined as follows:

```
typedef struct {
    Xrt3dDataType   type; /*XRT3D_DATA_IRGRID*/
    int             numx, numy;
    double          noval;
    double          *xgrid, *ygrid;
    double          **values;
}Xrt3dIrGridData;
```

The **Xrt3dGridData** structure contains a **type** field which is always initialized to **XRT3D_DATA_GRID** by **Xrt3dMakeIrGridData**() and **Xrt3dMakeDataFromFile**().

numx and *numy* are integers defining the number of grid lines (i.e. elements) in the X and Y dimensions. For surface and histogram charts, *numx* and *numy* must be ≥ 2. For bar charts, *numx* and *numy* must be ≥ 1.

The special value which Olectra Chart treats as “no value” is specified by *noval*. *xgrid* and *ygrid* are arrays of doubles of length *numx* and *numy* which contain the values of the X and Y grid lines.

values is a two-dimensional array of the Z values for each grid point in X major order.

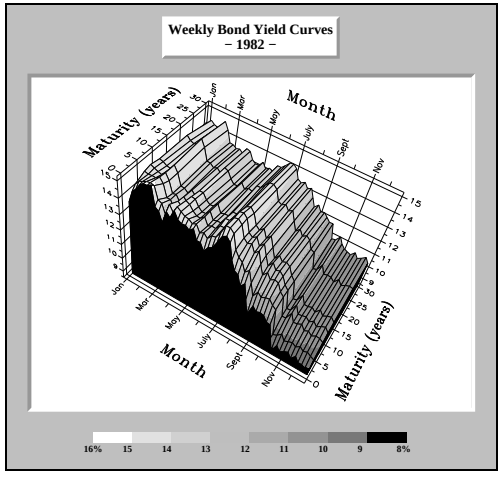


Figure 28 An Irregular Surface Plot

Example of Setting Values

An application can easily access the **Xrt3dData** structure to read or change values. The following example initializes a regular grid surface to 42.0:

```
int         i, j;
Xrt3dData   *data;

data = Xrt3dMakeGridData(...);

for (i = 0; i < data->g.numx; i++)  {
    for (j = 0; j < data->g.numy; j++) {
        data->g.values[i][j] = 42.0;
    }
}
```

Convenience Procedures

There are a number of **Xrt3dData** structure manipulation convenience procedures included with Olectra Chart. These procedures are fully described in Appendix C on page 99.

Xrt3dDataCopy()	Creates a copy of an **Xrt3dData** structure.
Xrt3dDataShaded()	Creates a new **Xrt3dData** structure from an existing structure and fills it with a shaded relief map of the original data.

Xrt3dDataSmooth()	Smooths the data in an **Xrt3dData** structure using center-weighted averaging.
Xrt3dDataWindow()	Creates a new **Xrt3dData** structure from an existing structure, extracting a subset of the original data and resampling it using either cubic or linear splines.

5

Programming User Interaction

Default User Interaction ■ *Overview of Action Maps and Messages*
Starting User Interaction ■ *Updating User Interaction*
Ending User Interaction ■ *Overview of Action Maps and Messages*
Interacting with Chart Data ■ *Window Resizing*

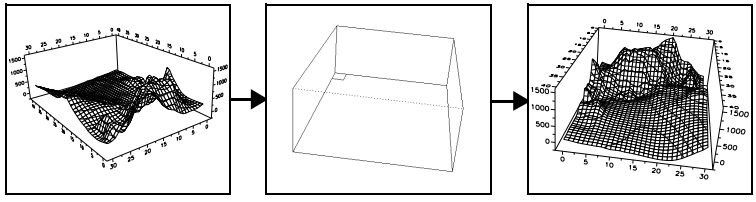
This chapter describes the user-interaction features of Olectra Chart–how a user can interact with the chart and how an application can control interaction.

5.1 Default User Interaction

By default, an end-user can rotate, translate, scale, and zoom into the unit cube. An application can also define action maps which manipulate a chart programmatically. Figure 29 shows the user interactions enabled by Olectra Chart's default action maps. Note that if you have a three-button mouse, holding down the middle mouse button is equivalent to simultaneously holding down the left and right mouse buttons.

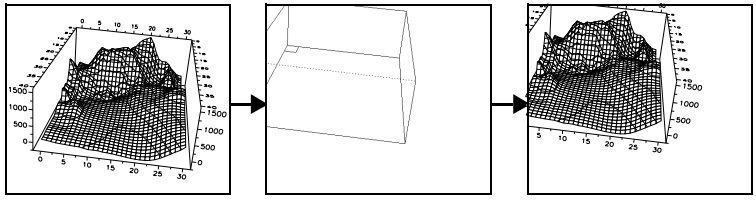
Rotation

- Hold down both mouse buttons and either:
- Move mouse counter-clockwise to rotate view clockwise *or*
- Press x, y, z, or e to select an axis, and then move mouse perpendicular to axis



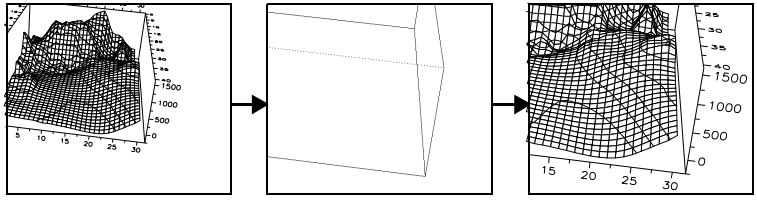
Translation

- Press Shift and hold down both mouse buttons
- Move mouse to shift the graph



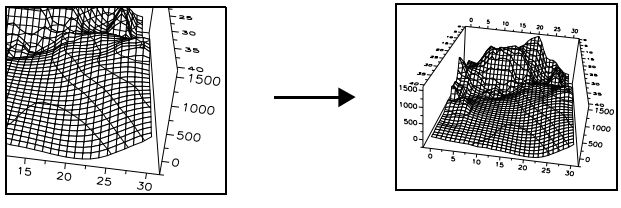
Scaling

- Press Ctrl and hold down both mouse buttons
- Move mouse down to zoom in
- Move mouse up to zoom out



Return to Default

- Press "r"
- All scaling, translation, and zooming removed



Zooming

- Press Ctrl and hold down left mouse button
- Move mouse to select the area to zoom into

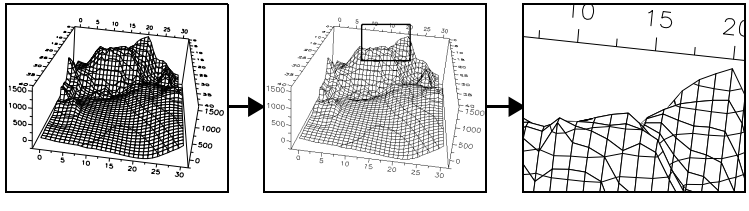


Figure 29 Olectra Chart's Default User Interactions

5.2 Overview of Action Maps and Messages

Olectra Chart's default action maps define user events that cause some interactive action within the control. You can customize user interaction through the following mechanisms:

- Action maps–An application can change or remove the default actions.
- Messages–An application can be notified as a user interacts with the control by defining message handler procedures that are called before, during, and after user interaction. A message handler procedure can affect each interaction in such ways as disallowing or constraining it. See the section on each interaction for details on using its callback.

Three Interaction Stages

Olectra Chart's default user interaction passes through three stages:

- Starting user interaction
- Updating user interaction, and
- Ending user interaction

An interaction must pass through these stages in sequence, and an application can be notified by messages during each stage. Each stage is described in the following sections.

5.3 Starting User Interaction

The **XRT3DN_MODIFY_START** message is passed to the window's message handler to notify the application that a user interaction is about to begin. Figure 30 illustrates this.

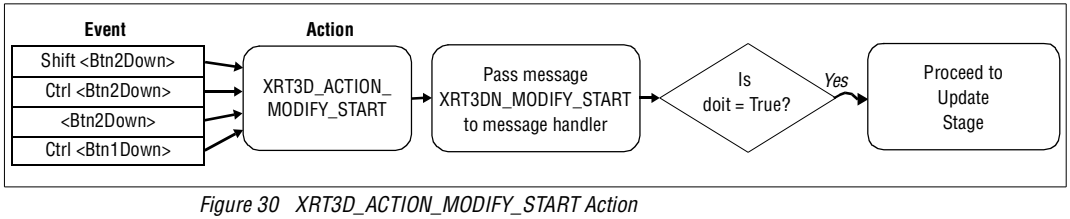


Figure 30 XRT3D_ACTION_MODIFY_START Action

Disabling All User Interaction

You can use the **XRT3DN_MODIFY_START** message to disable any user interaction regardless of the action maps installed. The window's message handler is passed the following message:

```
XRT3DN_MODIFY_START:
hWnd = (HWND) wParam;
mcb = (Xrt3dModifyCallbackStruct *) lParam;

typedef struct {
    BOOL doit;
} Xrt3dModifyCallbackStruct;
```

Set the *doit* parameter of this structure to **FALSE** to disable all user interactions. When *doit* is **FALSE**, all "update" actions are disabled until the next **XRT3DN_MODIFY_START** message is passed.

5.4 Updating User Interaction

One of several messages is passed to notify the application that a user interaction is about to be updated. No action can update the chart unless the interaction has successfully passed through the **XRT3D_ACTION_MODIFY_START** action.

Controlling Previewing

As a user rotates, scales, or translates a chart, Olectra Chart can provide user feedback either by drawing a quick outline of the unit cube or by drawing the entire unit cube. The **XRT3D_PREVIEW_METHOD** property controls this feedback, and can be set to either **XRT3D_PREVIEW_CUBE** (draw quick outline–the default) or **XRT3D_PREVIEW_FULL** (draw full unit cube). Drawing the entire unit cube works best with small datasets.

5.4.1 Scaling, Translation, and Zooming

The **XRT3D_ACTION_SCALE** action updates interactive scaling of the chart. The **XRT3D_ACTION_TRANSLATE** action updates interactive translation of the chart. The **XRT3D_ACTION_ZOOM_END** action zooms into the chart at the area defined by the "zoom rubberband" (defined by the **XRT3D_ACTION_ZOOM_UPDATE** action). These routines alter the **XRT3D_VIEW_SCALE** and **XRT3D_VIEW_[XY]TRANSLATE** properties. Figure 31 illustrates these action routines.

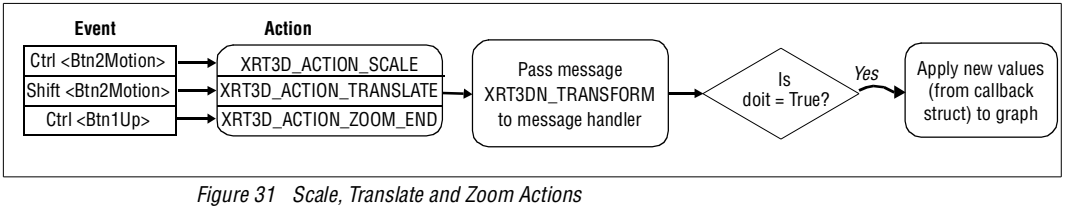


Figure 31 Scale, Translate and Zoom Actions

Controlling Interaction

You can use the **XRT3DN_TRANSFORM** message to control scaling, translation, or zooming. The following message is passed to the window's message handler:

```
XRT3DN_TRANSFORM:
hWnd = (HWND) wParam;
tcb = (Xrt3dTransformCallbackStruct *) lParam;

   typedef struct {
       double      scale;
       double      xtranslate;
       double      ytranslate;
       BOOL        doit;
   } Xrt3dTransformCallbackStruct;
```

A Transform action can change the *scale*, *xtranslate*, *ytranslate*, and *doit* parameters, which are then applied to the chart. For example, to constrain scaling, examine and change the *scale* parameter.

Resetting Interactions

The **XRT3D_ACTION_RESET** action removes all scaling and translation by restoring **XRT3D_VIEW_SCALE** and **XRT3D_VIEW_[XY]TRANSLATE** to their default values.

The **XRT3DN_TRANSFORM** message is sent to the window's message handler. An application can deny a reset by setting *doit* to **FALSE**.

5.4.2 Rotation

The **XRT3D_ACTION_ROTATE** action updates interactive rotation of the chart. This routine alters the **XRT3D_[XYZ]ROTATION** properties. Figure 32 illustrates this action routine.

XRT3D_VIEW_NORMALIZED

By default, Olectra Chart resizes the unit cube so that the entire chart is visible (including axis titles and annotation); this setting causes jerky interactive rotation. To allow a user to smoothly rotate a chart, **XRT3D_VIEW_NORMALIZED** should be set to **FALSE** before interactive rotation begins.

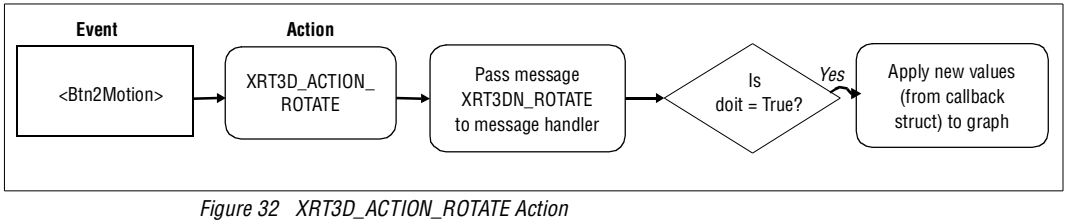


Figure 32 XRT3D_ACTION_ROTATE Action

Controlling Rotation

You can use the **XRT3D_ACTION_ROTATE** action to control rotation. The following message is sent to the window's message handler:

```
XRT3DN_ROTATE:
hWnd = (HWND) wParam;
tcb = (Xrt3dRotateCallbackStruct *) lParam;

   typedef struct {
       double      xrotation;
       double      yrotation;
       double      zrotation;
       BOOL        doit;
   } Xrt3dRotateCallbackStruct;
```

A Rotate action can change the *xrotation*, *yrotation*, *zrotation*, and *doit* parameters.

5.5 Ending User Interaction

The **XRT3D_ACTION_MODIFY_END** action redraws the control and notifies the application that a user interaction has finished. The message **XRT3DN_MODIFY_END** is sent:

```
XRT3DN_MODIFY_END:
hWnd = (HWND) wParam;
```

Note that no structure is passed.

5.6 Programming Actions

All Olectra Chart actions are customizable: you can determine which Microsoft Windows message should call a particular action, and decide on the appropriate steps to perform in each case.

Only mouse messages and the **WM_KEYDOWN** and **WM_KEYUP** messages are recognized.

5.6.1 Changing the Action Maps

An action map is a mapping of a particular Windows message to a predefined action. Each action map consists of three parts: the Windows message, any modifier flags, and the keycode (only if **WM_KEYDOWN** or **WM_KEYUP**).

The following messages are recognized:

WM_LBUTTONDBLCLK double-click left mouse button

WM_MBUTTONDBLCLK double-click both mouse buttons

WM_RBUTTONDBLCLK double-click right mouse button

WM_LBUTTONDOWN press left mouse button

WM_MBUTTONDOWN press both mouse buttons

WM_RBUTTONDOWN	press right mouse button
WM_LBUTTONUP	release left mouse button
WM_MBUTTONUP	release both mouse buttons
WM_RBUTTONUP	release right mouse button
WM_MOUSEMOVE	move mouse
WM_KEYDOWN	press key
WM_KEYUP	release key

Note that if you have a three-button mouse, holding down the middle mouse button is equivalent to simultaneously holding down the left and right mouse buttons.

Modifier Flags

The following modifier flags are recognized:

MK_LBUTTON	left mouse button
MK_MBUTTON	both mouse buttons
MK_RBUTTON	right mouse button
MK_ALT	Alt key
MK_SHIFT	Shift key
MK_CONTROL	Ctrl key

All actions are normalized to match the event sent by Microsoft Windows. For example, **MK_LBUTTON** is added to the modifier flags if a **WM_LBUTTONDOWN** message is sent.

Recognized Keycodes

Any valid **VK_** value is treated as a recognized keycode. Note the following, however:

- All alphabetic characters are forced to upper case.
- **MK_SHIFT** must appear in the modifier if capitals are desired.
- The CapsLock key toggles the meaning of the **MK_SHIFT** modifier.

Determining Action Mappings

To determine which action is mapped to a particular Microsoft Windows message, use the **Xrt3dGetAction**() function. For example, the following code determines which action is mapped to the left mouse button down message:

```
Xrt3dAction action;

action = Xrt3dGetAction(hXrt3d, WM_LBUTTONDOWN, 0, 0);
```

Any unmapped action returns **XRT3D_ACTION_NONE**.

To return the entire list of action maps, call **Xrt3dGetActionList**(). The pointer returned points to read-only memory.

Programming Action Mappings

To program an action mapping, call **Xrt3dSetAction**(). For example, the following code removes all previously defined actions:

```
Xrt3dActionItem *item, *next;

item = Xrt3dGetActionList(hXrt3d);
for (; item; item = next) {
    next = item->next;
    Xrt3dSetAction(hXrt3d, item->msg, item->modifier,
        item->keycode, XRT3D_ACTION_NONE);
}
```

Setting an action mapping to **XRT3D_ACTION_NONE** removes the action.

The following example uses the left mouse button plus the Alt key for rotation instead of both mouse buttons:

```
/* remove all use of both buttons */
Xrt3dSetAction(hXrt3d, WM_MBUTTONDOWN, 0, 0, XRT3D_ACTION_NONE);
Xrt3dSetAction(hXrt3d, WM_MOUSEMOVE, MK_MBUTTON, 0,
    XRT3D_ACTION_NONE);
Xrt3dSetAction(hXrt3d, WM_MBUTTONUP, 0, 0, XRT3D_ACTION_NONE);

/* reprogram for Alt+Left */
Xrt3dSetAction(hXrt3d, WM_LBUTTONDOWN, MK_ALT, 0,
    XRT3D_ACTION_MODIFY_START);
Xrt3dSetAction(hXrt3d, WM_MOUSEMOVE, MK_LBUTTON|MK_ALT, 0,
    XRT3D_ACTION_ROTATE);
Xrt3dSetAction(hXrt3d, WM_LBUTTONUP, MK_ALT, 0,
    XRT3D_ACTION_MODIFY_END);
```

5.6.2 Disabling and Disallowing Interactions

The easiest way to disallow interactions in Olectra Chart is to catch the **XRT3DN_MODIFY_START** message, and set the *doit* element in the passed structure to **FALSE**. Another approach is to remove all action mappings, as shown in the previous section.

To remove individual interactions, use **Xrt3dSetAction**() to set the desired interaction to **XRT3D_ACTION_NONE**.

5.6.3 Calling Actions Directly

To call a chart action directly, use **Xrt3dCallAction**(). This function expects four arguments:

- The graph handle
- The action to be called
- The X- and Y-coordinates of the window location at which the action is to be called

When an action is invoked, the window coordinates specified by **Xrt3dCallAction**() must be contained within the graph area.

5.7 Interacting with Chart Data

Microsoft Windows provides a mechanism for registering interest in events such as mouse movement, mouse button clicks, and keyboard clicks. An Olectra Chart program can make use of the event handling mechanisms to react to events that happen over a chart control.

Olectra Chart provides procedures for mapping the pixel coordinates of an event to:

- the indices of the grid point closest to the event coordinates.
- the exact chart coordinates at the event coordinates.

Xrt3dPick()

The **Xrt3dPick**() procedure takes a chart control handle, (x,y) pixel coordinates and a pointer to an **Xrt3dPickResult** structure. It fills in the **Xrt3dPickResult** structure with information about the grid point closest to the specified pixel coordinates on the specified chart control:

```
Xrt3dRegion
Xrt3dPick(graph, pix_x, pix_y, pick)
    HXRT3D              graph;
    int                 pix_x, pix_y;
    Xrt3dPickResult     *pick;
```

The **Xrt3dPickResult** structure is defined as:

```
typedef struct {
    int         pix_x, pix_y;
    int         xindex, yindex;
    int         distance;
} Xrt3dPickResult;
```

The fields are broken down as follows:

pix_x The x pixel coordinate passed to **Xrt3dPick**().

pix_y The y pixel coordinate passed to **Xrt3dPick**().

xindex The index of the X grid line closest to the pixel coordinates.

yindex The index of theY grid line closest to the pixel coordinates.

distance The screen distance (in pixels) between the given pixel coordinates and the on-screen display of the data value $Z_{(xindex,\ yindex)}$.

Xrt3dPick() returns one of the following values:

XRT3D_RGN_NOWHERE The given pixel coordinates are not close enough to anything to be picked. *xindex* and *yindex* are set to -1.

XRT3D_RGN_IN_GRAPH The given pixel coordinates are in the graph area. *xindex* and *yindex* and *distance* are set correctly. If a hole in the data is picked, *xindex* and *yindex* are set to -1.

XRT3D_RGN_IN_LEGEND The given pixel coordinates are in the legend area. *xindex* and *yindex* are set to -1.

XRT3D_RGN_IN_HEADER The given pixel coordinates are not in the graph area, but are in the header area. *xindex* and *yindex* are set to -1.

XRT3D_RGN_IN_FOOTER The given pixel coordinates are not in the graph area, but are in the footer area. *xindex* and *yindex* are set to -1.

Xrt3dUnpick()

The **Xrt3dUnpick**() procedure is the opposite of **Xrt3dPick**(). It determines the pixel coordinate displaying a grid point. See Appendix C on page 99 for more details.

```
WM_LBUTTONDOWN:
    int     x, y;
    Xrt3dPickResult p;

    x = LOWORD(lParam);
    y = HIWORD(lParam);
    Xrt3dPick(hXrt3d, x, y, &pick);
```

Figure 33 Message handler calling Xrt3dPick()

Xrt3dMap()

The **Xrt3dMap**() procedure takes a chart control handle, (x,y) pixel coordinates and a pointer to a map structure. It maps the center of the pixel coordinates to a point on the surface. Surface values are obtained by interpolating from the nearest grid values. It writes the pixel coordinates and floating-point chart coordinates into the map structure.

```
Xrt3dRegion
Xrt3dMap(graph, pix_x, pix_y, map)
    HXRT3D          graph;
    int             pix_x, pix_y;
    Xrt3dMapResult  *map;
```

The **Xrt3dMapResult** structure is defined as:

```
typedef struct {
    int         pix_x, pix_y;
    double      x, y, z;
} Xrt3dMapResult;
```

The fields are broken down as follows:

pix_x The X coordinate passed into **Xrt3dMap**().

pix_y The Y coordinate passed into **Xrt3dMap**().

x The mapped chart coordinate X-value.

y The mapped chart coordinate Y-value.

z The mapped chart coordinate Z-value.

Xrt3dMap() returns **XRT3D_RGN_NOWHERE**, **XRT3D_RGN_IN_FOOTER**, **XRT3D_RGN_IN_HEADER**, **XRT3D_RGN_IN_LEGEND** or **XRT3D_RGN_IN_GRAPH**. If the mapped pixel is not on the surface, the map result's *x*, *y* and *z* elements are all set to **XRT3D_HUGE_VAL**.

Xrt3dUnmap()

The **Xrt3dUnmap**() procedure is the opposite of **Xrt3dMap**(). It maps from chart coordinates to pixel coordinates. See Appendix C on page 99 for more details.

Xrt3dComputeZValue()

The value of a grid cell can be modified programmatically using **Xrt3dComputeZValue**(). This function is passed the indices of the grid cell to be modified and the new pixel location of the point.

This procedure can be used to support the interactive modification of a grid value. For example: when the user clicks somewhere on the surface, call **Xrt3dPick**() to determine the closest grid index. Then, as the mouse is dragged, new pixel values are passed along with the grid indices to **Xrt3dComputeZValue**(), which returns the new Z value for that grid cell. When the drag has completed, place the new Z value into the appropriate grid cell, and assign the updated data structure to the property **XRT3D_SURFACE_DATA**. See the *bars* demo program and source code for an example of this type of user interaction.

Mapping/Picking Bar Charts

Xrt3dMap() and **Xrt3dPick**() can be used on bar charts as well as 3D surface charts. If passed the coordinates of a pixel that is not part of any bar, **Xrt3dMap**() will return **XRT3D_HUGE_VAL** in the map result's *x*, *y* and *z* elements; **Xrt3dPick**() will return **-1** in the pick result's *x* and *y* indices.

For pixels that are used to display any portion of the top, bottom or sides of a bar, **Xrt3dMap**() returns the actual X, Y and Z values of the pixel.

5.8 Window Resizing

Most applications should allow the user to resize a window containing a chart control and have the control adjust to its new window size.

To resize the chart when the user resizes the window containing it, change the chart control's size in a **WM_SIZE** message handler, for example:

```
case WM_SIZE:
{
    int width = LOWORD(lParam);
    int height = HIWORD(lParam);

    SetWindowPos(hwndXrt3D, NULL, 0, 0, width, height,
            SWP_NOMOVE | SWP_NOZORDER);
    break;
}
```

Repaint & Resize Messages

An application can find out when the chart control is repainted or resized by checking for the **XRT3DN_REPAINTED** or **XRT3DN_RESIZED** notification messages. These messages contain information in the **lParam** parameter. See Appendix D on page 121 for complete details. The resize message is sent after the chart is resized. The repaint message is sent after the chart is redrawn.

An application can use these messages to draw onto the chart image using Windows API functions. Another use is to adjust the control properties, depending on the size of the chart control.

The following message handler uses **XRT3DN_RESIZED** to remove the legend from the chart when it gets too small:

```
case XRT3DN_RESIZED:
    cb = (Xrt3dCallbackStruct *) lParam;
    if (cb->width <= 300) {
        Xrt3dSetValues(hChart,
            XRT3D_LEGEND_SHOW, FALSE,
            NULL);
    } else {
        Xrt3dSetValues(hChart,
            XRT3D_LEGEND_SHOW, TRUE,
            NULL);
    }
    break;
```

6

Advanced Olectra Chart Programming

4D Surface Charts ■ *4D Bar Charts*
Text Objects ■ *Customizing the Distribution Table*
Customizing Legend Labels ■ *Customizing Contour Styles*

This section covers topics that programmers of advanced Olectra Chart applications will find useful. It assumes that you are already familiar with Olectra Chart.

6.1 4D Surface Charts

Olectra Chart can be used to display 4D charts using color as a fourth dimension. The additional color information is provided to Olectra Chart as a second **Xrt3dData** structure using the **XRT3D_ZONE_DATA** property.

To create a 4D chart:

- Set DrawZones and DrawShaded to **TRUE**.
- Using the **XRT3D_SURFACE_DATA** property, attach data to provide the 3D surface shape.
- Attach data to be used for deriving the zoning and contouring colors as **XRT3D_ZONE_DATA**.
- Make sure both **Xrt3dData** structures have the same origin and the same number of X and Y points. If regularly gridded data is being used, then the *xstep* and *ystep* must be identical in both structures. If irregularly-gridded data is being used, the values of *xgrid* and *ygrid* must be identical in both structures. It is not possible to use a regularly gridded data set for the surface data, and an irregularly gridded data for the zone data, or vice versa. If any of these conditions are not met, a 4D chart will not be displayed.

If the zone data has a hole that is not in the surface data, the surface in the region of the hole will be displayed as if the zone data were not attached.

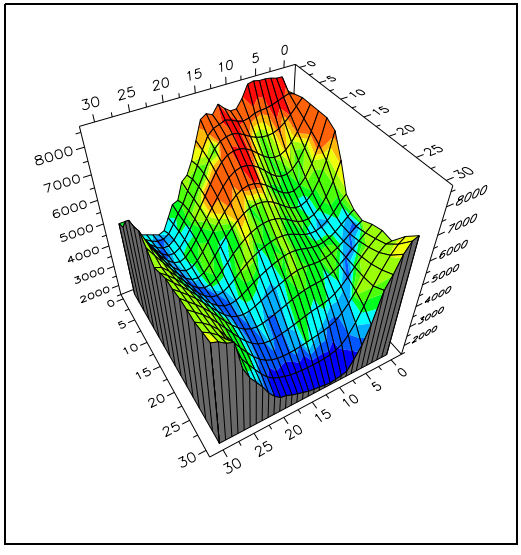


Figure 34 4D chart—Zone/Contour data is different from Surface data

6.2 4D Bar Charts

When **XRT3D_ZONE_DATA** is supplied for a bar chart, the values in the zone data are used in conjunction with the distribution table to apply zone colors to the bars in the grid.

When zone data is supplied and DrawZones is **TRUE**, the bar is not broken up into separate colored segments. Rather, each bar is individually colored according to the zoned height of the bar. Contours are never drawn when zone data is supplied. Figure 35 gives an example of a 4D bar chart.

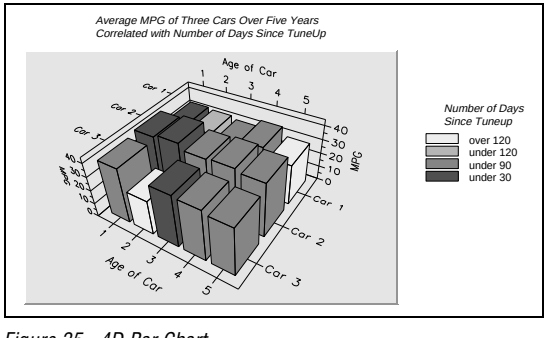


Figure 35 4D Bar Chart

In a bar chart, the zone data structure is only referenced when DrawZones is **TRUE**. A legend is generated based on the distribution table. The legend labels can be replaced, or supplied in a property with the **XRT3D_LEGEND_STRINGS** property. Section 6.5 on page 71 has more details on supplying legend strings.

6.3 Text Objects

A surface or bar chart can contain one or more *text objects*. A text object is an independent rectangular region that can be attached to the chart in one of three ways:

- To a particular grid index,
- To a point in 3D space, or
- To a set of pixel coordinates on the control.

You can use text objects in a number of ways, such as annotating points of interest on a surface or as "headers" that appear inside the unit cube.

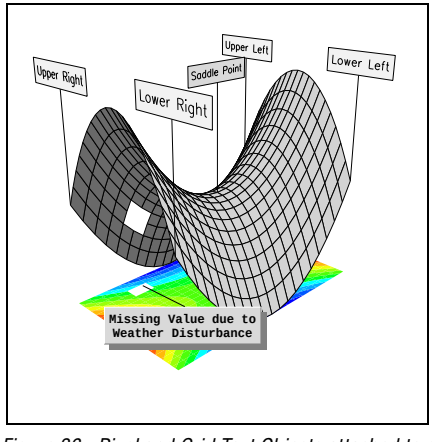


Figure 36 Pixel and Grid Text Objects attached to a surface chart

Text Object Types

The **XRT3D_TEXT_ATTACH_METHOD** property specifies how a text object is attached to the chart–this defines the type of text object it is. A *Grid* text object is specified when attachment is **XRT3D_ATTACH_INDEX** or **XRT3D_ATTACH_POINT**. A Pixel text object is specified when attachment is **XRT3D_ATTACH_PIXEL**. A Grid text object displayed on a 2D chart (that is, a surface with DrawMesh and DrawShaded set to **FALSE**) is referred to as a 2D Grid text object. Figure 37 shows the properties that apply to text objects.

The following points summarize the characteristics of each text object type:

- 2D Grid text objects–attached to the unit cube itself; drawn over the chart; scale and translate with the chart; use stroke fonts.

- 3D Grid text objects–attached to the unit cube itself (either to an X-Y grid point or a point in 3D space); drawn from back to front; rotate, scale, and translate with the chart; use stroke fonts.
- Pixel text objects–similar to a header or footer; attached at a set of control pixel coordinates; drawn over the chart; stay stationary when the chart rotates, scales or translates; use X fonts; have a customizable border.

Note: Multiple text objects that intersect are not rendered correctly by Olectra Chart. Applications should ensure that text objects are arranged so that they do not intersect each other.

Common Properties

XRT3D_TEXT_ADJUST	XRT3D_TEXT_ATTACH_METHOD	XRT3D_TEXT_BACKGROUND_COLOR
XRT3D_TEXT_FOREGROUND_COLOR	XRT3D_TEXT_LINE_SHOW	XRT3D_TEXT_OFFSET_[XY] †
XRT3D_TEXT_PRINT_FONT	XRT3D_TEXT_SHOW	XRT3D_TEXT_STRINGS

† XRT3D_TEXT_OFFSET_[XY] does not apply to 3D Grid text objects

Grid-only Properties

XRT3D_TEXT_ATTACH_INDEX_[XY]	XRT3D_TEXT_ATTACH_POINT_[XYZ]	XRT3D_TEXT_PLANE
XRT3D_TEXT_STROKE_FONT	XRT3D_TEXT_STROKE_SIZE	

Pixel-only Properties

XRT3D_TEXT_ATTACH_PIXEL_[XY]	XRT3D_TEXT_BORDER	XRT3D_TEXT_BORDER_WIDTH
XRT3D_TEXT_FONT		

Figure 37 Text Object Property Summary

Creating a Text Object

To add a text object to a chart, create it as a child of the chart, as shown by the following example:

```
HXRT3DTEXT text_obj;
char *strings[] = {"Hello World!", NULL};
text_obj = Xrt3dTextCreate(graph);
Xrt3dTextSetValues(text_obj, XRT3D_TEXT_STRINGS, strings, NULL);
```

Text

The text displayed in a text object is specified with the **XRT3D_TEXT_STRINGS** property. Text objects can display multiple lines of text.

The alignment of multi-line text is specified with the **XRT3D_TEXT_ADJUST** property. Text can be left-aligned (**XRT3D_ADJUST_LEFT**), centered (**XRT3D_ADJUST_CENTER**), or right-aligned (**XRT3D_ADJUST_RIGHT**).

3D Grid Object Positioning

To attach a text object to a grid index, set **XRT3D_TEXT_ATTACH_METHOD** to **XRT3D_ATTACH_INDEX** and **XRT3D_TEXT_ATTACH_INDEX_[XY]** to an X and Y index on the chart. The X and Y values must be within the X and Y range of the dataset.[1] Olectra Chart draws the text object above the maximum Z value on the surface. The offset properties are ignored but the plane the text object is oriented to can be specified using **XRT3D_TEXT_PLANE**.

To attach a text object to a point in 3D space, set **XRT3D_TEXT_ATTACH_METHOD** to **XRT3D_ATTACH_POINT** and **XRT3D_TEXT_ATTACH_POINT_[XYZ]** to an X, Y, and Z point in 3D. These values must be within the data range. If the Z value is less than the Z value on the surface, Olectra Chart draws the text object below the minimum Z value; if it is greater or equal to the dataset's Z value (or equal to the hole value), the text object is drawn above the maximum Z value. The offset properties are ignored but the plane the text object is oriented to can be specified using **XRT3D_TEXT_PLANE**.

The following example positions a 3D Grid text object:

```
Xrt3dTextSetValues(my_text,
    XRT3D_TEXT_ATTACH_METHOD,   XRT3D_ATTACH_POINT,
    XRT3D_TEXT_ATTACH_POINT_X,  20.0,
    XRT3D_TEXT_ATTACH_POINT_Y,  20.0,
    XRT3D_TEXT_ATTACH_POINT_Z,  500.0,
    NULL);
```

2D Grid Object Positioning

Text objects on 2D surfaces are attached the same way as 3D surfaces (**XRT3D_TEXT_ATTACH_POINT_Z** is ignored on text objects attached to a point in 3-D space). Olectra Chart draws these text objects on top of the chart in the order they were created.

Use the **XRT3D_TEXT_OFFSET_[XY]** properties to move the text object away from its point of attachment.

Pixel Object Positioning

To attach a text object to pixel coordinates on the control, set **XRT3D_TEXT_ATTACH_METHOD** to **XRT3D_ATTACH_PIXEL** and **XRT3D_TEXT_ATTACH_PIXEL_[XY]** to a set of (x,y) pixel coordinates. Olectra Chart draws these text objects in the order they were created, after any Grid text objects.

Use the **XRT3D_TEXT_OFFSET_[XY]** properties to move the text object away from its point of attachment.

Connecting Line

You can place a line between a text object and its attachment point using the **XRT3D_TEXT_LINE_SHOW** property. This property should be used with the

1. If the Z-value at the X and Y index is a hole, no text object is drawn because the hole is considered to be out of data range.

XRT3D_TEXT_OFFSET_[XY] properties to set a distance between the text object and the attachment point.

Font

Grid text objects use **XRT3D_TEXT_STROKE_FONT** and **XRT3D_TEXT_STROKE_SIZE** to specify the font used. See Appendix E on page 125 for a list of the valid stroke fonts.

Pixel text objects use **XRT3D_TEXT_FONT** to specify the font used. This property is specified the same as header, footer, and legend fonts. See section 3.15 on page 40 for further details.

Border

Pixel text objects can be enhanced with a border using the **XRT3D_TEXT_BORDER** and **XRT3D_TEXT_BORDER_WIDTH** properties. Any valid Olectra Chart border type may be used; see section 3.17 on page 42 for details on Olectra Chart borders.

Grid text objects do not use this property. They have a single-pixel border drawn in the foreground color of the text object. If the background color is **NULL**, no border appears.

Colors

You can set a background and foreground color for a text object using **XRT3D_TEXT_BACKGROUND** and **XRT3D_TEXT_FOREGROUND**.

Text Object Performance

Applications should batch the creation or manipulation of multiple text objects. See section 2.5 on page 15 for more information on batching.

Removing a Text Object

To remove a text object from the display, set **XRT3D_TEXT_SHOW** to **FALSE**. To permanently destroy a text object, call **Xrt3dTextDestroy**().

6.4 Customizing the Distribution Table

By default, Olectra Chart will use a linear distribution table with **XRT3D_NUM_DISTN_LEVELS** when DrawContours and/or DrawZones is **TRUE**.

To specify your own distribution table, set **XRT3D_DISTN_METHOD** to **XRT3D_DISTN_FROM_TABLE** and then supply your own distribution table using the **XRT3D_DISTN_TABLE** property.

XRT3D_DISTN_TABLE specifies a pointer to an **Xrt3dDistnTable** structure:

```
typedef struct {
    int     nentries;
    double  *entry;
} Xrt3dDistnTable;
```

nentries tells Olectra Chart how many entries are in the table. *entry* is an array of doubles specifying the boundaries of each level.

When a distribution table is defined, *nentries* defines the number of distribution levels *nlevels*, and **XRT3D_NUM_DISTN_LEVELS** is ignored. Recall that to display *nlevels* of distribution, *nlevels* contour line styles are needed, and the fill color from the *(nlevel + 1)*th contour style is needed for the last (highest) zone color when DrawZones is **TRUE**.

For example, the following code defines a distribution table which specifies 2 levels. The first fill color displays all values ≤ -5. The second style displays all values > -5, and ≤ 40. The third style's zone fill color displays all values > 40.

```
static double e[] = { -5.0, 40.0 };
static Xrt3dDistnTable mytable;

mytable.nentries = 2;
mytable.entry = e;

Xrt3dSetValues(mygraph,
    XRT3D_DISTN_METHOD, XRT3D_DISTN_FROM_TABLE,
    XRT3D_DISTN_TABLE,  &mytable,
    NULL);
```

6.5 Customizing Legend Labels

There are two ways of specifying your own legend labels. You can define your own labeling function, or you can specify labels for each level in the distribution table. Each method is described below.

Legend Label Function

To specify your own legend labels, set **XRT3D_LEGEND_LABEL_FUNC** to the name of your own label generating function.

Whenever Olectra Chart needs to generate a legend label, it will call your function. Your function must then return a label to use. The form of the function is:

```
char *
MyLegendFunc(widget, level, zmin, zmax, label)
HXRT3D  widget;
int     level;
double  zmin, zmax;
char *  label;
```

control is the handle of the chart control. *level* is the (zero-indexed) level that a label is being generated for. *zmin* and *zmax* define the level's lowest and highest values. *label* is the string that Olectra Chart would generate by default.

To return to default labels, set **XRT3D_LEGEND_LABEL_FUNC** to **NULL**.

The *label* string can contain the special characters "**\l**", "**\c**" and "**\r**" for left, center and right field justification. Each field is preceded by one of these special characters. Remember to escape the "\" itself with another "\". If no special characters are provided, left justification is assumed.

For example, the following code generates normal legend labels, except when the level includes Z=212. It will label this single level as "Boiling Zone":

```
char *
mylabfunc(widget, level, zmin, zmax, label)
HXRT3D  widget;
int     level;
double  zmin, zmax;
char *  label;
{
    if (zmin < 212.0 && 212.0 <= zmax)
       return("\\cBoiling Zone")
    else
       return(label);
}

Xrt3dSetValues(mygraph;
    XRT3D_LEGEND_LABEL_FUNC, mylabfunc,
    NULL);
```

If the legend is stepped, and your label function returns **NULL**, Olectra Chart will skip the legend entry altogether.

Legend Label Strings

Another way to get custom legend labels is to supply a list of strings to the **XRT3D_LEGEND_STRINGS** property. These strings are used to annotate each level in the distribution table. This method is most useful when displaying a 4D bar chart where groups of bars with the same color require a common label.

The actual strings used can still be overridden programmatically using the **XRT3D_LEGEND_LABEL_FUNC** property.

6.6 Customizing Contour Styles

By default, Olectra Chart provides an array of 100 contour styles. The array used for the chart is specified by the **XRT3D_CONTOUR_STYLES** property.You will need to provide custom contour styles if:

- it is important to your application to specify the precise contour style for any particular level;
- if you want to display more than 100 levels; or
- if you want to uniquely identify contour lines. The Olectra Chart default contour styles use only black solid lines of width 1.

It is usually easiest to specify more contour styles than will be needed for the number of levels of distribution. If *nstyles* contour styles are provided, and the number of distribution levels is *nlevels*, Olectra Chart will calculate the index into the contour styles array for level i ($0 \le i \le nlevels$) as follows[1]:

$$\lfloor i \times (nstyles - 1) / nlevels \rfloor$$

Xrt3dContourStyle

Each contour style is defined by an **Xrt3dContourStyle** structure (declared in the OLCH3DCM.H header file):

```
typedef struct {
    char    *fill_color;        /*for zoning*/
    char    *line_color;        /*line color*/
    int     line_width;         /*line width*/
    Xrt3dLinePattern lpat;      /*2D contours only*/
} Xrt3dContourStyle;
```

The **Xrt3dContourStyle** data structure contains information about how Olectra Chart should display contours and zones in a set of data. The fields are broken down as follows:

fill_color The color used to demarcate the level when DrawZones is **TRUE**. Use the **RGB**() macro to specify this color.

line_color The color used to demarcate the level's contour line when DrawContours is **TRUE**. Use the **RGB**() macro to specify this color.

line_width The line width used to demarcate the level's contour line when DrawContours is **TRUE**. Must be greater than or equal to 0. When **line_width** is zero, no line is drawn.

1. $\lfloor x \rfloor$ means floor(x); that is, the largest integer less than or equal to x.

lpat The line pattern used to demarcate the level's contour line when DrawContours is **TRUE**. Line patterns are only honored when viewing in 2D (i.e. when DrawMesh and DrawShaded are both **FALSE** and when **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**). The line pattern must be one of the **XRT3D_LPAT_** constants listed in Figure 38.

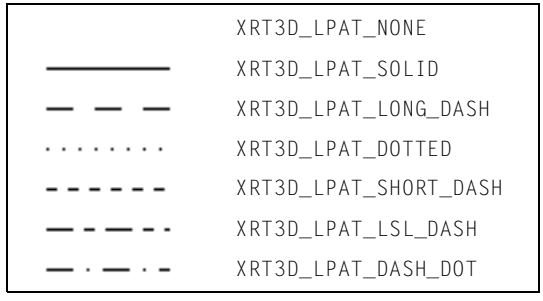


Figure 38 Line Patterns

Explicit Contour Styles

If you do not want to use Olectra Chart's default contour styles, you may set your own contour styles.

Contour styles are accessed through the **XRT3D_CONTOUR_STYLES** property and/or through the **Xrt3dSetNthContourStyle**() and **Xrt3dGetNthContourStyle**() procedures.

XRT3D_CONTOUR_STYLES can be used to get or set the entire array of contour styles. When used with **Xrt3dGetValues**(), the returned array pointer should be considered read-only. The **Xrt3dDupContourStyles**() procedure may be used to duplicate the returned contour styles array pointer. For example, to set the first and second contour style's line widths to 3 pixels thick, the following code could be used:

```
Xrt3dContourStyle **cs, **mycs;
Xrt3dGetValues(myg, XRT3D_CONTOUR_STYLES, &cs,
               NULL);
mycs = Xrt3dDupContourStyles(cs);
mycs[0]->line_width = 3;
mycs[1]->line_width = 3;
Xrt3dSetValues(myg,
    XRT3D_CONTOUR_STYLES, mycs,
    NULL);
Xrt3dFreeContourStyles(mycs);
```

A program can get a pointer to a particular **Xrt3dContourStyle** structure using **Xrt3dGetNthContourStyle**(). For example, the following code will print out the 50th contour style's line thickness and double it:

```
char buffer[100];

Xrt3dContourStyle*cstyle, my_cstyle;
cstyle = Xrt3dGetNthContourStyle(myg, 49,FALSE);
sprintf(buffer, "line width for 50th CStyle: %d",
    cstyle->line_width);
MessageBox(hWnd, buffer, "Information",  MB_OK);
my_cstyle = * cstyle;
my_cstyle.line_width = my_cstyle.line_width * 2;
Xrt3dSetNthContourStyle(myg, 49,
            &my_cstyle, FALSE);
```

Contour Style Tips

To return to using default contour styles, call the **Xrt3dResetContourStyles**() procedure.

The number of zoning or contouring levels, *nlevels*, is defined by **XRT3D_NUM_DISTN_LEVELS**. (If a distribution table has been provided, then *nlevels* is the number of entries in the table.) If DrawContours is **TRUE** and DrawZones is **FALSE**, *nlevels* of contour styles will be needed. If DrawZones is **TRUE**, *nlevels + 1* contour styles will be needed, as the $(nlevels+1)^{th}$ contour style will just be used for defining the last (i.e. highest) zone's fill color.

If you want to have a one-to-one mapping from the supplied contour styles to the styles used on screen, make sure that you supply exactly *nlevels + 1* contour styles.

If too few contour styles are provided for the number of levels requested, Olectra Chart will re-sample the supplied contour styles.

Part II

Reference Appendices

A

Olectra Chart Property Reference

Control Synopsis
Olectra Chart Properties

This appendix lists all of the Olectra Chart properties in alphabetical order. Listed after the property name are its data type and default value.

A.1 Control Synopsis

Include File: \INCLUDE\OLCH3D.H

Class Name: "OlectraChart3D"

A.2 Olectra Chart Properties

XRT3D_AXIS_STROKE_FONT **Xrt3dStrokeFont** **XRT3D_SF_ROMAN_SIMPLEX**

Specifies the font to be used for the axis annotation. May be set to any of the valid stroke fonts listed in Appendix E.

XRT3D_AXIS_STROKE_SIZE **int** **80**

Specifies the size of the axis annotation font. The size is measured in thousandths of the unit cube size, and must be between 0 and 1000. See section 2.1 on page 11 for a description of the unit cube.

XRT3D_AXIS_TITLE_STROKE_FONT **Xrt3dStrokeFont** **XRT3D_SF_ROMAN_SIMPLEX**

Specifies the font to be used for the axis titles. May be set to any of the valid stroke fonts listed in Appendix E.

XRT3D_AXIS_TITLE_STROKE_SIZE **int** **80**

Specifies the size of the axis title font. The size is measured in thousandths of the unit cube size, and must be between 0 and 1000. See section 2.1 on page 11 for a description of the unit cube.

XRT3D_BACKGROUND_COLOR **COLORREF** **RGB(255,255,255)**

Specifies the window background color. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification. This color will be inherited as the default background color for the chart, data area, header, footer and legend.

XRT3D_BORDER **Xrt3dBorder (enum)** **XRT3D_BORDER_NONE**

Specifies the style of border used to enclose the chart. Valid styles are **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW**, **XRT3D_BORDER_PLAIN**.

XRT3D_BORDER_WIDTH **int** **2**

Specifies the width of the chart border in pixels. Must be between 0 and 20.

XRT3D_CONTOUR_STYLES **Xrt3dContourStyle** ****dynamic***

The **Xrt3dContourStyle** structure defines a fill color for zoning, and a line color, line width and line style for contouring.

Use **XRT3D_CONTOUR_STYLES** to determine what contour styles are currently defined, or to define new styles to be used. To set new contour styles, create a **NULL**-terminated array of contour style structure pointers, and set **XRT3D_CONTOUR_STYLES** to be a pointer to this array. When used with **Xrt3dGetValues**(), a pointer to an array of **Xrt3dContourStyle** structures is returned.

The number of styles actually used, *nlevels*, depends on **XRT3D_DISTN_METHOD** and **XRT3D_NUM_DISTN_LEVELS**. If a linear distribution method is being used, then *nlevels* is the value of **XRT3D_NUM_DISTN_LEVELS**. If a distribution table is being used, then *nlevels* is the number of table entries.

If more than enough contour styles have been provided, Olectra Chart will evenly sample the provided styles. Specifically, if *nstyles* contour styles have been provided, the index of the contour style used at each level i is:[1]

$$\lfloor i \times (nstyles - 1) / nlevels \rfloor$$

If not enough contour styles are defined, Olectra Chart will re-sample the provided contour styles as needed.

When contour styles are not specified by **XRT3D_CONTOUR_STYLES**, Olectra Chart will use default contour styles. The default contour styles all have 100 entries. To return to using

1. $\lfloor x \rfloor$ means floor(x); that is, the largest integer less than or equal to x.

the default contour styles after having set **XRT3D_CONTOUR_STYLES**, call the **Xrt3dResetContourStyles**() procedure.

The procedures **Xrt3dDupContourStyles**() and **Xrt3dFreeContourStyles**(), and the methods **Xrt3dGetNthContourStyle**() and **Xrt3dSetNthContourStyle**() may be used to manipulate contour style structures.

XRT3D_DATA_AREA_BACKGROUND_COLOR COLORREF XRT3D_DEFAULT_COLOR

Specifies the background color of the chart's data area. When **XRT3D_DEFAULT_COLOR**, the data area background is transparent. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification. This property is ignored in pie charts.

XRT3D_DEBUG BOOL FALSE

Specifies whether to send warning messages to a debug window. When set to **FALSE**, only property conversion errors are output.

XRT3D_DISTN_METHOD Xrt3dDistnMethod (enum) XRT3D_DISTN_LINEAR

Specifies the type of distribution method used. Must be one of **XRT3D_DISTN_LINEAR** or **XRT3D_DISTN_FROM_TABLE**. Setting this property to **XRT3D_DISTN_FROM_TABLE** will only have effect if a distribution table has been provided (using **XRT3D_DISTN_TABLE**).

XRT3D_DISTN_TABLE Xrt3dDistnTable * NULL

Use this property to specify a custom distribution. The *nentries* element in the **Xrt3dDistnTable** structure specifies the number of entries in the table. *entry* is a pointer to an array of doubles specifying the distribution separation.

```
typedef struct {
    int         nentries;
    double      *entry;
} Xrt3dDistnTable;
```

If **XRT3D_DISTN_METHOD** is set to **XRT3D_DISTN_FROM_TABLE**, and a distribution table has been supplied, it will be used to define the distribution. All surface data values less than or equal to the first entry will be graphed in the first zone and displayed using the first contour style. Surface data values greater than the n-1th entry and less than or equal to the nth entry will be graphed in the nth zone. Surface values greater than the last entry will be graphed in the *nentries*+1th zone, and displayed using the last contour style.

Entries that are outside of the chart's Z-range are not displayed in the legend. Entries do not have to be specified in ascending order because Olectra Chart will sort the entries before using them. Olectra Chart will also cull any duplicate entries.

XRT3D_DOUBLE_BUFFER BOOL TRUE

When **TRUE**, display updates are first rendered into an off-screen bitmap, then copied to the display area. This reduces flashing on the screen. Setting this property to **FALSE** will cause chart images to be rendered directly to the display, resulting in less memory utilization.

XRT3D_DRAW_CONTOURS BOOL FALSE

When **TRUE**, Olectra Chart will draw contour lines identifying each entry in the distribution. The contour line color, line width and line pattern are determined by the value of

XRT3D_CONTOUR_STYLES. The distribution is determined by **XRT3D_DISTN_METHOD**, **XRT3D_DISTN_TABLE** and **XRT3D_NUM_DISTN_LEVELS**. This property is ignored when a 4D bar chart is drawn.

If DrawContours is **TRUE** and DrawZones is **FALSE**, Olectra Chart will automatically generate a legend entry for each contour line.

This property is used in conjunction with the DrawMesh, DrawShaded and DrawZones properties to define the 15 basic chart types. See section 3.2 on page 24 for more information.

XRT3D_DRAW_HIDDEN_LINES **BOOL** **FALSE**

This property determines if mesh, contour and axis lines, hidden from the observer's view, should be drawn. For realistic images, this boolean should be set to **FALSE**. Rendering performance improves when set to **TRUE**.

This property has no effect when both DrawMesh and DrawContours are **FALSE**, or when either DrawShaded or DrawZones is **TRUE**.

XRT3D_DRAW_MESH **BOOL** **TRUE**

When **TRUE**, and when **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**, Olectra Chart will display data on a three-dimensional mesh. The mesh lines in either the X or Y direction may be turned off using **XRT3D_[XY]MESH_SHOW**. Display of hidden mesh lines is controlled with **XRT3D_DRAW_HIDDEN_LINES**. The color of the top and bottom of the mesh is specified with **XRT3D_MESH_BOTTOM_COLOR** and **XRT3D_MESH_TOP_COLOR**.

When **TRUE**, and when **XRT3D_TYPE** is **XRT3D_TYPE_BAR**, Olectra Chart will draw the outline of each bar. All bars whose value is above the value of **XRT3D_ZORIGIN** will be outlined using the **XRT3D_MESH_TOP_COLOR**; those bars with value below the value of **XRT3D_ZORIGIN** will be outlined using the color **XRT3D_MESH_BOTTOM_COLOR**.

Display of hidden lines can be controlled with **XRT3D_DRAW_HIDDEN_LINES**.

This property is used in conjunction with the DrawShaded, DrawContours and DrawZones properties to define the 15 basic chart types. See section 3.2 on page 24 for more information.

XRT3D_DRAW_SHADED **BOOL** **FALSE**

When **TRUE**, Olectra Chart will use flat shading to display the 3D surface (when **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**) and the bar surfaces (when **XRT3D_TYPE** is **XRT3D_TYPE_BAR**). The color of the top and bottom of the 3D surface is specified with **XRT3D_SURFACE_TOP_COLOR** and **XRT3D_SURFACE_BOTTOM_COLOR**. For bar charts, these colors are used to indicate whether a bar is above or below the value of **XRT3D_ZORIGIN**.

This property is used in conjunction with the DrawMesh, DrawContours and DrawZones properties to define the 15 basic chart types. See section 3.2 on page 24 for more information.

XRT3D_DRAW_ZONES **BOOL** **FALSE**

When **TRUE**, Olectra Chart will fill the zones (areas between either contour lines or cells) identifying each entry in the distribution. The zone color is determined by the value of **XRT3D_CONTOUR_STYLES**. The distribution is determined by **XRT3D_DISTN_METHOD**, **XRT3D_DISTN_TABLE** and **XRT3D_NUM_DISTN_LEVELS**. Normally, **XRT3D_SURFACE_DATA** is used to determine zones; however, if **XRT3D_ZONE_DATA** is

defined, it will be used instead, resulting in a 4D display. See sections 6.1 and 6.2 for more information on 4D surfaces and bar charts.

When DrawZones is **TRUE**, Olectra Chart will automatically generate a legend entry for each zone color.

This property is used in conjunction with the DrawMesh, DrawShaded and DrawContours properties to define the 15 basic chart types. See section 3.2 on page 24 for more information.

XRT3D_FOOTER_ADJUST **Xrt3dAdjust (enum)** **XRT3D_ADJUST_CENTER**

Specifies how multiple footer text lines should be adjusted within the footer area. Must be one of **XRT3D_ADJUST_CENTER**, **XRT3D_ADJUST_LEFT** or **XRT3D_ADJUST_RIGHT**.

XRT3D_FOOTER_BACKGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the footer area background color. When **XRT3D_DEFAULT_COLOR**, the footer background is transparent. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_FOOTER_BORDER **Xrt3dBorder (enum)** **XRT3D_BORDER_NONE**

Specifies the style of border used to identify the footer area. Valid styles are **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW**, **XRT3D_BORDER_PLAIN**.

XRT3D_FOOTER_BORDER_WIDTH **int** **2**

Specifies the width of the footer area border in pixels. Must be between 0 and 20.

XRT3D_FOOTER_FONT **HFONT** **Arial, 12 pt**

Specifies the font to use for the footer strings. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT3D_FOOTER_FOREGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the footer area foreground color. When **XRT3D_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_FOOTER_STRINGS **char **** **NULL**

Specifies a **NULL**-terminated list of strings to be displayed in the footer area.

XRT3D_FOOTER_WIDTH **int** ***dynamic***
XRT3D_FOOTER_HEIGHT **int** ***dynamic***

Contains the height and width of the footer area in pixels.

XRT3D_FOOTER_X **int** **centered under chart area**
XRT3D_FOOTER_Y **int** **centered under chart area**

Specifies the x or y pixel offset of the top left corner of the footer, relative to the top left corner of the window. For example, (10, 70) will cause the footer to begin 10 pixels to the right of and

70 pixels down from the top left corner of the window. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT3D_FOOTER_X_USE_DEFAULT **BOOL** **TRUE**
XRT3D_FOOTER_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT3D_FOOTER_X** and **XRT3D_FOOTER_Y** values are determined by Olectra Chart at render-time. When explicit **XRT3D_FOOTER_X** and **XRT3D_FOOTER_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**.

XRT3D_FOREGROUND_COLOR **COLORREF** **RGB(0,0,0)**

Specifies the window foreground color. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification. This color will be inherited as the default foreground color for the header, footer, and legend text, as well as the chart axes.

XRT3D_GRAPH_BACKGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the chart area background color. When **XRT3D_DEFAULT_COLOR**, the chart is transparent. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_GRAPH_BORDER **Xrt3dBorder (enum)** **XRT3D_BORDER_NONE**

Specifies the style of border used to identify the chart area. Valid styles are **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW**, **XRT3D_BORDER_PLAIN**.

XRT3D_GRAPH_BORDER_WIDTH **int** **2**

Specifies the width of the chart area border in pixels. Must be between 0 and 20.

XRT3D_GRAPH_FOREGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the chart area foreground color. When **XRT3D_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_GRAPH_WIDTH **int** ***dynamic***
XRT3D_GRAPH_HEIGHT **int** ***dynamic***

Specifies the height and width of the chart area in pixels. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT3D_GRAPH_WIDTH_USE_DEFAULT **BOOL** **TRUE**
XRT3D_GRAPH_HEIGHT_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT3D_GRAPH_WIDTH** and **XRT3D_GRAPH_HEIGHT** values are determined by Olectra Chart at render-time. When explicit **XRT3D_GRAPH_WIDTH** and **XRT3D_GRAPH_HEIGHT** values are provided, Olectra Chart sets these Booleans to **FALSE**.

XRT3D_GRAPH_X **int** **derived from data**
XRT3D_GRAPH_Y **int** **derived from data**

Specifies the x or y pixel offset of the top left corner of the chart area, relative to the top left corner of the window. For example, (10, 5) will cause the chart area to begin 10 pixels to the right, and 5 pixels down from the top left corner of the window. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT3D_GRAPH_X_USE_DEFAULT **BOOL** **TRUE**
XRT3D_GRAPH_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT3D_GRAPH_X** and **XRT3D_GRAPH_Y** values are determined by Olectra Chart at render-time. When explicit **XRT3D_GRAPH_X** and **XRT3D_GRAPH_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**. You cannot set these properties to **FALSE** unless you have previously provided **XRT3D_GRAPH_X** or **XRT3D_GRAPH_Y** values.

XRT3D_HEADER_ADJUST **Xrt3dAdjust (enum)** **XRT3D_ADJUST_CENTER**

Specifies how multiple header text lines should be adjusted within the header area. Must be one of **XRT3D_ADJUST_CENTER**, **XRT3D_ADJUST_LEFT** or **XRT3D_ADJUST_RIGHT**.

XRT3D_HEADER_BACKGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the header area background color. When **XRT3D_DEFAULT_COLOR**, the header is transparent. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_HEADER_BORDER **Xrt3dBorder (enum)** **XRT3D_BORDER_NONE**

Specifies the style of border used to identify the header area. Valid styles are **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW**, **XRT3D_BORDER_PLAIN**.

XRT3D_HEADER_BORDER_WIDTH **int** **2**

Specifies the width of the header area border in pixels. Must be between 0 and 20.

XRT3D_HEADER_FONT **HFONT** **Arial, 12 pt**

Specifies the font to use for the header strings. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT3D_HEADER_FOREGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the header area foreground color. When **XRT3D_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_HEADER_STRINGS **char **** **NULL**

Specifies a **NULL**-terminated list of strings to be displayed in the header area.

XRT3D_HEADER_WIDTH **int** ***dynamic***
XRT3D_HEADER_HEIGHT **int** ***dynamic***

Contains the height and width of the header area in pixels.

NOTE: These properties cannot be set.

XRT3D_HEADER_X **int** **centered above chart area**
XRT3D_HEADER_Y **int** **centered above chart area**

Specifies the x or y pixel offset of the top left corner of the header, relative to the top left corner of the window. For example, (10, 20) will cause the header to begin 10 pixels to the right of, and 20 pixels down from, the top left corner of the window. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT3D_HEADER_X_USE_DEFAULT **BOOL** **TRUE**
XRT3D_HEADER_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT3D_HEADER_X** and **XRT3D_HEADER_Y** values are determined by Olectra Chart at render-time. When explicit **XRT3D_HEADER_X** and **XRT3D_HEADER_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**.

XRT3D_HEIGHT **int** **size of created window**

Specifies the height of the control window.

XRT3D_LEGEND_ANCHOR **Xrt3dAnchor (enum)** **XRT3D_ANCHOR_EAST**

Specifies where to anchor the legend to the window. Valid values are: **XRT3D_ANCHOR_NORTH**, **XRT3D_ANCHOR_SOUTH**, **XRT3D_ANCHOR_EAST**, **XRT3D_ANCHOR_WEST**, **XRT3D_ANCHOR_NORTHWEST**, **XRT3D_ANCHOR_NORTHEAST**, **XRT3D_ANCHOR_SOUTHWEST**, **XRT3D_ANCHOR_SOUTHEAST**.

XRT3D_LEGEND_BACKGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the legend area background color. When **XRT3D_DEFAULT_COLOR**, the legend background is transparent. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_LEGEND_BORDER **Xrt3dBorder (enum)** **XRT3D_BORDER_NONE**

Specifies the style of border used to identify the legend area. Valid styles are **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW**, **XRT3D_BORDER_PLAIN**.

XRT3D_LEGEND_BORDER_WIDTH **int** **2**

Specifies the width of the legend area border in pixels. Must be between 0 and 20.

XRT3D_LEGEND_FONT **HFONT** **Arial, 12 pt**

Specifies the font to be used in the legend. If the Arial TrueType font is not available on the system, the System font is used. If you are using Microsoft Windows 3.1, cast the value to an **int** when setting this property.

XRT3D_LEGEND_FOREGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the legend area foreground color. When **XRT3D_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_LEGEND_LABEL_FUNC **Function** **NULL**

This property can be used to provide legend labels. If not **NULL**, Olectra Chart will call the supplied function to obtain a label for each legend level (starting from 0). The form of the function is:

```
char * MyLegendFunction(hChart, level, zmin, zmax, label)
    HXRT3D      hChart;
    int         level;
    double      zmin, zmax;
    char *      label;
```

Once a specific function is provided, Olectra Chart will call it, passing as parameters the *control* handle, the *level* for which a label is required (zero-indexed), the *zmin* and *zmax* that define the level, and the *label* that Olectra Chart would normally calculate by default. This function should return a string that Olectra Chart will use for that level's legend label. If the legend is stepped, and this function returns **NULL**, Olectra Chart will skip the legend entry.

The label string can contain up to thirty fields which are delimited by **'\l'**, **'\c'** and **'\r'** for *left*, *center* and *right* justification. For example, "**\r32.0\cto\r39.6**". Remember to escape the **'\'** with another **'\'**.

XRT3D_LEGEND_ORIENTATION **Xrt3dAlign (enum)** **XRT3D_ALIGN_VERTICAL**

Specifies the orientation of the legend. Valid values are **XRT3D_ALIGN_HORIZONTAL** and **XRT3D_ALIGN_VERTICAL**.

XRT3D_LEGEND_SHOW **BOOL** **TRUE**

Determines if the legend should be displayed. Legends are created automatically by Olectra Chart whenever contours or zones are drawn.

XRT3D_LEGEND_STRINGS **char **** **NULL**

Specifies strings to be used in the legend. The number of distribution levels is determined by the value of **XRT3D_NUM_DISTN_LEVELS** or **XRT3D_DISTN_TABLE**, and the setting of **XRT3D_LEGEND_STYLE**. If too few strings are provided, the corresponding entries in the legend are dropped. A legend is generated only when either contours or zones are drawn, and **XRT3D_LEGEND_SHOW** is **TRUE**.

These legend strings override anything that the control would normally have generated; however, if an **XRT3D_LEGEND_LABEL_FUNC** has been specified, the legend strings may be overridden dynamically at runtime. This property is most often used to annotate particular groups of bars in a 4D bar chart.

XRT3D_LEGEND_STYLE **Xrt3dLegendStyle** **XRT3D_LEGEND_STYLE_CONTINUOUS**

Determines the legend style and must be one of **XRT3D_LEGEND_STYLE_STEPPED** or **XRT3D_LEGEND_STYLE_CONTINUOUS**. Stepped legends have the range printed for each zone. Continuous legends have the level printed at the division between two adjacent zones, and are more compact.

XRT3D_LEGEND_WIDTH **int** ***dynamic***
XRT3D_LEGEND_HEIGHT **int** ***dynamic***

Contains the height and width of the legend area in pixels.

NOTE: These properties cannot be set.

XRT3D_LEGEND_X **int** **depends on anchor**
XRT3D_LEGEND_Y **int** **depends on anchor**

Specifies the (X,Y) coordinates of the top left corner of the legend area. A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT3D_LEGEND_X_USE_DEFAULT **BOOL** **TRUE**
XRT3D_LEGEND_Y_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, **XRT3D_LEGEND_X** and **XRT3D_LEGEND_Y** values are determined by Olectra Chart at render-time. When explicit **XRT3D_LEGEND_X** and **XRT3D_LEGEND_Y** values are provided, Olectra Chart sets these Booleans to **FALSE**.

XRT3D_MESH_BOTTOM_COLOR **COLORREF** **RGB(0,0,0)**
XRT3D_MESH_TOP_COLOR **COLORREF** **RGB(0,0,0)**

Specifies the color of the 3D surface mesh lines or bar chart outlines displayed when **XRT3D_DRAW_MESH** is **TRUE**. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

For bar charts, the top color is used for bars with a value greater than **XRT3D_ZORIGIN**, and the bottom color is used for bars with a value less than **XRT3D_ZORIGIN**.

XRT3D_NAME **char *** **dynamic**

Specifies the name of a particular chart instance. By default, Olectra Chart generates a unique name for each chart created.

XRT3D_NUM_DISTN_LEVELS **int** **10**

Specifies the number of distribution levels (i.e. *nlevels*) used when DrawContours and/or DrawZones is **TRUE**. Any value between 0 and 100 is valid. Setting it to 0 causes no contour lines to be drawn, and if DrawZones is **TRUE**, only one zone is drawn. If a custom distribution table has been provided, and **XRT3D_DISTN_METHOD** is **XRT3D_DISTN_FROM_TABLE**, the number of distribution levels is the number of entries in the table and **XRT3D_NUM_DISTN_LEVELS** is ignored.

XRT3D_PERSPECTIVE_DEPTH **double** **2.5**

Controls the perspective effect. Range is 1 to **XRT3D_HUGE_VAL**. Small values exaggerate perspective while large values reduce it. To reduce the perspective effect and approach a parallel projection, set this property to a very high number (such as 100,000).

XRT3D_PREVIEW_METHOD **Xrt3dPreviewMethod** **XRT3D_PREVIEW_CUBE**

Controls how the interactive rotations are previewed to the user. **XRT3D_PREVIEW_CUBE** draws only the outline of the cube. **XRT3D_PREVIEW_FULL** draws the entire chart during rotations (useful for small charts or bar charts).

XRT3D_PROJECT_ZMAX **int** **0**
XRT3D_PROJECT_ZMIN **int** **0**

Specifies whether to project contours and/or zones on the *z=zmin* or *z=zmax* plane. These properties are specified using a combination of the following constants:

```
#define XRT3D_PROJECT_CONTOURS  0x1
#define XRT3D_PROJECT_ZONES     0x2
```

Contours/zones are projected regardless of the setting of DrawContours and DrawZones. These properties are ignored in a 2D chart, and when **XRT3D_TYPE** is **XRT3D_TYPE_BAR.**

For example, to project both contours and zones, set the property to:

```
XRT3D_PROJECT_CONTOURS | XRT3D_PROJECT_ZONES
```

XRT3D_REPAINT **BOOL** **TRUE**

When this property is **TRUE**, any changes made to the control are rendered immediately. In order to batch changes to the control, set this property to **FALSE**, make the changes, and then on the last **Xrt3dSetValues**() call, set this property to **TRUE** again.

XRT3D_SOLID_SURFACE **BOOL** **FALSE**

When this property is **TRUE**, and DrawMesh or DrawShaded is **TRUE**, Olectra Chart will render the 3D surface as a solid object by adding "sides" to the surface. The sides drop from the surface to the minimum Z value. The mesh and surface bottom colors are used when drawing the "sides". This property has no effect on bar charts.

XRT3D_SURFACE_BOTTOM_COLOR **COLORREF** **RGB(112,112,112)**
XRT3D_SURFACE_TOP_COLOR **COLORREF** **RGB(211,211,211)**

Specifies the color of the bottom and top of the 3D surface and bar chart facets displayed when **XRT3D_DRAW_SHADED** is **TRUE**. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

For bar charts, the top color is used for bars with a value greater than **XRT3D_ZORIGIN**, and the bottom color is used for bars with a value less than **XRT3D_ZORIGIN**.

XRT3D_SURFACE_DATA **Xrt3dData *** **NULL**

Specifies the data to be displayed in 3D surface charts and bar charts. The **Xrt3dData** structure should be allocated using the procedure **Xrt3dMakeGridData**() or **Xrt3dMakeDataFromFile**(). See section 4.2 on page 49 for information on this structure.

XRT3D_TYPE **Xrt3dType** **XRT3D_TYPE_SURFACE**

Specifies the basic type of chart that will be drawn. It may be **XRT3D_TYPE_SURFACE** or **XRT3D_TYPE_BAR**. Many variations can be displayed by manipulating the four major Boolean properties. See section 3.2 on page 24 for more information.

XRT3D_VIEW_NORMALIZED **BOOL** **TRUE**

When **TRUE**, Olectra Chart always calculates the full view (before taking into account any view scaling or translation) so as to maximize the view without causing any portion of the view to be clipped. Dynamically rotating when this property is **TRUE** will look "jerky". To eliminate "jerky rotation", set this property to **FALSE**.

XRT3D_VIEW_SCALE **double** **1.0**

Specifies the degree of scaling (i.e. zooming) up or down from a full view (i.e. a maximal, non-clipped view). Valid values must be between 1/30 and 30.0. This property is updated if the user dynamically scales or zooms their view. See section 5.4.1 on page 56 for information on dynamic scaling.

XRT3D_VIEW_XTRANSLATE **double** **0.0**
XRT3D_VIEW_YTRANSLATE **double** **0.0**

Specifies the degree of view translation from a full view (i.e. a maximal, non-clipped view). The translations are relative to a normalized box, with origin in the bottom left corner. Translation of more than a full unit (i.e. more than $\pm$ 1.0) will cause the view to translate "out of view". This property is updated if the user dynamically translates their view. See section 5.4.1 on page 56 for information on dynamic translation.

XRT3D_WIDTH **int** **size of created window**

Specifies the width of the control window.

XRT3D_XANNO_METHOD **Xrt3dAnnoMethod** **XRT3D_ANNO_VALUES**
XRT3D_YANNO_METHOD **Xrt3dAnnoMethod** **XRT3D_ANNO_VALUES**
XRT3D_ZANNO_METHOD **Xrt3dAnnoMethod** **XRT3D_ANNO_VALUES**

Specifies the method used to annotate the axis. It is an enumerated type, with possible values **XRT3D_ANNO_VALUES**, **XRT3D_ANNO_DATA_LABELS**, and **XRT3D_ANNO_VALUE_LABELS**. If **XRT3D_ANNO_VALUES**, the strings are generated by the control automatically, based on the range of the data. If **XRT3D_ANNO_DATA_LABELS**, the value of the property **XRT3D_[XY]DATA_LABELS** is used; this method is not available for the Z-axis. If **XRT3D_ANNO_VALUE_LABELS**, the value of the property **XRT3D_[XYZ]VALUE_LABELS** is used.

XRT3D_XAXIS_SHOW **BOOL** **TRUE**
XRT3D_YAXIS_SHOW **BOOL** **TRUE**
XRT3D_ZAXIS_SHOW **BOOL** **TRUE**

Determines if the X-, Y-, or Z-axis should be displayed.

XRT3D_XAXIS_TITLE **char *** **NULL**
XRT3D_YAXIS_TITLE **char *** **NULL**
XRT3D_ZAXIS_TITLE **char *** **NULL**

Specifies the title to display on the X-, Y- and Z-axis.

XRT3D_XBAR_FORMAT **Xrt3dBarFormat** **XRT3D_BAR_FIXED**
XRT3D_YBAR_FORMAT **Xrt3dBarFormat** **XRT3D_BAR_FIXED**

Specifies the representation format of the bar chart along each of the gridded axes. When set to **XRT3D_BAR_FIXED**, the bars in the chart are centered about their X values and Y values, and each bar is a fixed width. When set to **XRT3D_BAR_HISTOGRAM**, each bar fills the space up to its adjoining bar, and the width of each bar is derived from the separation between adjacent lines in the grid.

XRT3D_XBAR_SPACING **double** **80.0**
XRT3D_YBAR_SPACING **double** **80.0**

Specifies the amount of space used to display bars (in the X or Y direction) as a percentage of the total amount of space available. Values between 1.0 and 100.0 are valid. A value of 100.0 will cause the bars to abut one another if using regular data (they may not abut if using irregular data). This property only applies when **XRT3D_TYPE** is set to **XRT3D_TYPE_BAR**, and **XRT3D_[XY]BAR_FORMAT** is **XRT3D_BAR_FIXED**.

XRT3D_XDATA_LABELS **char **** **NULL**
XRT3D_YDATA_LABELS **char **** **NULL**

This property is used to supply the labels that will be used when the **XRT3D_[XY]ANNO_METHOD** is set to **XRT3D_ANNO_DATA_LABELS**. This is a **NULL**-terminated list of strings, where each label is applied to a line of data in the grid.

XRT3D_XGRID_LINES **int** **0**
XRT3D_YGRID_LINES **int** **0**
XRT3D_ZGRID_LINES **int** **0**

These properties can be used to specify grid lines on each applicable plane. The three planes are represented by constant values, and combined together to specify the affected planes exactly. In 2D, any non-zero value causes grid lines to be drawn perpendicular to the axis. The grid lines are drawn wherever an axis label would be drawn. The following constants are defined in the OLCH3DCM.H header file, and should be used when setting the value of these properties:

```
#define XRT3D_XY_PLANE 1
#define XRT3D_XZ_PLANE 2
#define XRT3D_YZ_PLANE 4
```

For example, to draw X grid lines on all possible planes, set **XRT3D_XGRID_LINES** to:

```
XRT3D_XY_PLANE | XRT3D_XZ_PLANE
```

To remove grid lines, set these properties to zero.

XRT3D_XMAX **double** ***dynamic***
XRT3D_YMAX **double** ***dynamic***
XRT3D_ZMAX **double** ***dynamic***

Specifies the value of the high end of the X, Y, and Z data displayed. X and Y data values higher than this value are not displayed. To zoom into the data, set narrower X and Y minimums and maximums. **XRT3D_ZMAX** cannot be less than the maximum Z data value.[1] A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

1. Olectra Chart sets the corresponding **USE_DEFAULT** property to **TRUE** when the value of this property falls inside the Z-range.

XRT3D_XMAX_USE_DEFAULT **BOOL** **TRUE**
XRT3D_YMAX_USE_DEFAULT **BOOL** **TRUE**
XRT3D_ZMAX_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, the value of **XRT3D_[XYZ]MAX** is determined by Olectra Chart at render time. When an explicit value for **XRT3D_[XYZ]MAX** is provided, Olectra Chart sets this property to **FALSE**. You cannot set this property to **FALSE** until a value for **XRT3D_ZMAX** has been calculated or provided. If the value of **XRT3D_ZMAX** is determined to be less than the maximum Z-value of the data, the value of **XRT3D_ZMAX_USE_DEFAULT** is forced to **TRUE**.

XRT3D_XMESH_FILTER **int** **1**
XRT3D_YMESH_FILTER **int** **1**

Specifies how the mesh is filtered before being displayed. For a value *n*, every *nth* mesh line is drawn. When set to 0, Olectra Chart automatically determines a pleasing mesh filter. The filter applies to both regular and irregular grids.

These properties affects the display only when DrawMesh is **TRUE**. They are ignored when the value of **XRT3D_TYPE** is **XRT3D_TYPE_BAR**.

XRT3D_XMESH_SHOW **BOOL** **TRUE**
XRT3D_YMESH_SHOW **BOOL** **TRUE**

Specifies whether Olectra Chart should draw the mesh lines in the X or Y direction when **XRT3D_DRAW_MESH** is **TRUE**. These properties have no effect on bar charts.

XRT3D_XMIN **double** ***dynamic***
XRT3D_YMIN **double** ***dynamic***
XRT3D_ZMIN **double** ***dynamic***

Specifies the value of the low end of the X, Y, and Z data displayed. X and Y data values lower than this value are not displayed. To zoom into the data, set narrower X and Y minimums and maximums. **XRT3D_ZMIN** cannot be greater than the minimum Z data value.[1] A side-effect of setting these properties is that the corresponding **USE_DEFAULT** property is set to **FALSE**.

XRT3D_XMIN_USE_DEFAULT **BOOL** **TRUE**
XRT3D_YMIN_USE_DEFAULT **BOOL** **TRUE**
XRT3D_ZMIN_USE_DEFAULT **BOOL** **TRUE**

When **TRUE**, the value of **XRT3D_[XYZ]MIN** is determined by Olectra Chart at render time. When an explicit value for **XRT3D_[XYZ]MIN** is provided, Olectra Chart sets this property to **FALSE**. You cannot set this property to **FALSE** until a value for **XRT3D_ZMIN** has been calculated or provided. If the value of **XRT3D_ZMIN** is determined to be greater than the minimum Z-value of the data, the value of **XRT3D_ZMIN_USE_DEFAULT** is forced to **TRUE**.

1. Olectra Chart sets the corresponding **USE_DEFAULT** property to **TRUE** when the value of this property falls inside the Z-range.

XRT3D_XROTATION **double** **45.0**
XRT3D_YROTATION **double** **0.0**
XRT3D_ZROTATION **double** **45.0**

Specifies the number of degrees of rotation of the unit cube, in a counter-clockwise direction about the axis. Olectra Chart applies rotations in Z-Y-X order. These properties are updated if the user dynamically rotates their view. See section 5.4.2 on page 57 for information on dynamic rotations.

XRT3D_XSCALE **double** **1.0**
XRT3D_YSCALE **double** **1.0**
XRT3D_ZSCALE **double** **1.0**

The **XRT3D_[XYZ]SCALE** properties can be used to scale the unit cube. For example, to scale the unit cube up by a factor of two in the Y direction (resulting in an oblong unit cube, twice as long in the Y direction as in the X and Z directions), set **XRT3D_YSCALE** to 2.0.

XRT3D_XVALUE_LABELS **Xrt3dValueLabel **** **NULL**
XRT3D_YVALUE_LABELS **Xrt3dValueLabel **** **NULL**
XRT3D_ZVALUE_LABELS **Xrt3dValueLabel **** **NULL**

This property is used to specify labels for the axis to be used when **XRT3D_[XY]ANNO_METHOD** is **XRT3D_ANNO_VALUE_LABELS**. These labels provide a set of strings that are applied at specific values along the axis. See section 3.12 on page 38 for more information.

XRT3D_XY_COLORS **Xrt3dXYColor **** **NULL**

Specifies a list of colors that are to be applied, in order, to bar charts when **XRT3D_DRAW_SHADED** is **TRUE** and **XRT3D_DRAW_ZONES** is **FALSE**. These colors can be applied to an entire row or column of bar charts, or to an individual cell. See the description of the **Xrt3dXYColor** structure in section 3.3 on page 29 for more information.

This property has no effect when **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**.

XRT3D_ZONE_DATA **Xrt3dData *** **NULL**

When **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**, this property specifies the zone (i.e. filled contour) data to be displayed draped over surface data (which is specified with **XRT3D_SURFACE_DATA**) in a 4D chart. This property should be set to **NULL** unless projections or a 4D chart is desired. See section 6.1 on page 65 for information on 4D surface charts.

When **XRT3D_TYPE** is **XRT3D_TYPE_BAR**, this property specifies a dataset which is used to determine bar colors, resulting in a 4D bar chart. See section 6.2 for more information on 4D bar charts.

XRT3D_ZONE_METHOD **Xrt3dZoneMethod** **XRT3D_ZONE_CONTOURS**

Specifies the method used to fill the zone regions, displayed when DrawZones is **TRUE**. When set to **XRT3D_ZONE_CONTOURS**, Olectra Chart fills between each contour line. When set to **XRT3D_ZONE_CELLS**, Olectra Chart fills entire cells in a surface grid, based on the average value in the cell.

XRT3D_ZORIGIN **double** **0.0**

Defines the Z-value from which bars are filled when **XRT3D_TYPE** is **XRT3D_TYPE_BAR**. This property has no effect on the chart when **XRT3D_TYPE** is **XRT3D_TYPE_SURFACE**.

B

Olectra Chart Text Object Property Reference

This appendix lists all of the Olectra Chart text object properties in alphabetical order. Listed after the property name is its data type and default value.

XRT3D_TEXT_ADJUST **Xrt3dAdjust** **XRT3D_ADJUST_CENTER**

Specifies how multiple lines of text should be aligned within the text object. Valid values are **XRT3D_ADJUST_LEFT**, **XRT3D_ADJUST_CENTER** and **XRT3D_ADJUST_RIGHT**.

XRT3D_TEXT_ATTACH_INDEX_X **int** **0**
XRT3D_TEXT_ATTACH_INDEX_Y **int** **0**

Specifies the grid index that the text object is attached to. This property affects text object positioning when **XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_INDEX**.

XRT3D_TEXT_ATTACH_METHOD **Xrt3dTextAttachMethod** **XRT3D_TEXT_ATTACH_INDEX**

Specifies how the text object is attached to the chart. Valid values are **XRT3D_TEXT_ATTACH_PIXEL**, **XRT3D_TEXT_ATTACH_POINT** and **XRT3D_TEXT_ATTACH_INDEX**. The positioning of the text object is controlled by properties that correspond to the attachment method: **XRT3D_TEXT_ATTACH_PIXEL_[XY]**, **XRT3D_TEXT_ATTACH_POINT_[XYZ]**, and **XRT3D_TEXT_ATTACH_INDEX_[XY]**.

XRT3D_TEXT_ATTACH_PIXEL_X **int** **0**
XRT3D_TEXT_ATTACH_PIXEL_Y **int** **0**

Specifies the control pixel coordinates that the text object is attached to. This property affects text object positioning when **XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_PIXEL**.

XRT3D_TEXT_ATTACH_POINT_X **double** **0.0**
XRT3D_TEXT_ATTACH_POINT_Y **double** **0.0**
XRT3D_TEXT_ATTACH_POINT_Z **double** **0.0**

Specifies the 3D chart coordinates that the text object is attached to. This property affects text object positioning when **XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_POINT**.

XRT3D_TEXT_BACKGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the text object background color. When **XRT3D_DEFAULT_COLOR**, the legend background is transparent. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_TEXT_BORDER **Xrt3dBorder** **XRT3D_BORDER_NONE**

Specifies the style of border used to identify the text object. This property is only used when the text object is attached by pixel (**XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_PIXEL**). Valid styles are **XRT3D_BORDER_NONE**, **XRT3D_BORDER_3D_IN**, **XRT3D_BORDER_3D_OUT**, **XRT3D_BORDER_ETCHED_IN**, **XRT3D_BORDER_ETCHED_OUT**, **XRT3D_BORDER_SHADOW**, **XRT3D_BORDER_PLAIN**.

Text objects attached by point or index have a 1-pixel line around the object, displayed when the text object background color is not **NULL**.

XRT3D_TEXT_BORDER_WIDTH **int** **2**

Specifies the width of the text object border when **XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_PIXEL**. Must be between 0 and 20.

XRT3D_TEXT_FONT **HFONT** **Arial, 12 pt**

Specifies the font used for the text contained in the text object. This property is used when **XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_PIXEL**. **XRT3D_TEXT_STROKE_FONT** specifies the font used when the text object is attached by point or index. If the Arial TrueType font is not available on the system, the System font is used.

XRT3D_TEXT_FOREGROUND_COLOR **COLORREF** **XRT3D_DEFAULT_COLOR**

Specifies the text object foreground color. When **XRT3D_DEFAULT_COLOR**, the window foreground color is used. The value is a Windows color reference. See section 3.18 on page 43 for details of color specification.

XRT3D_TEXT_LINE_SHOW **BOOL** **TRUE**

Specifies whether to draw a connecting line from the text object to the point of attachment. This is a pixel-wide line drawn in the foreground color of the text object.

XRT3D_TEXT_OFFSET_X **int** **0**
XRT3D_TEXT_OFFSET_Y **int** **0**

Specifies the pixel offset between the center of the text object and the point of attachment. This property is only used when the text object is either attached to a pixel, or the Olectra Chart control is in a 2D projection.

XRT3D_TEXT_PLANE **int** **XRT3D_XZ_PLANE**

Specifies the orientation of the text object. Text objects must be drawn parallel to one the planes of the cube, and are rendered so that the text is always drawn left-to-right. Valid values are **XRT3D_XZ_PLANE** and **XRT3D_YZ_PLANE**. This property is ignored in a 2D projection.

XRT3D_TEXT_SHOW **BOOL** **TRUE**

Specifies whether to display the text object.

XRT3D_TEXT_STRINGS **char **** **NULL**

Specifies a **NULL**-terminated list of strings displayed in the text object.

XRT3D_TEXT_STROKE_FONT **Xrt3dStrokeFont** **XRT3D_SF_ROMAN_SIMPLEX**

Specifies the stroke font used for the text contained in the text object. This property is used when **XRT3D_TEXT_ATTACH_METHOD** is **XRT3D_TEXT_ATTACH_POINT** or **XRT3D_TEXT_ATTACH_INDEX**. This property can be set to any of the valid stroke fonts listed in Appendix E on page 130. **XRT3D_TEXT_FONT** specifies the X font used when the text object is attached by pixel.

XRT3D_TEXT_STROKE_SIZE **int** **80**

Specifies the size of the stroke font. The size is measured in thousandths of the unit cube size, and must be between 0 and 1000. See page 11 for a description of the unit cube.

C

Procedures and Methods Reference

This appendix lists the Olectra Chart procedures and methods in alphabetical order.

Xrt3dAttachWindow()

Attaches a window to a chart. Returns **True** if the attach was successful.

```
BOOL
Xrt3dAttachWindow(
    HXRT3D      hChart,
    HWND        hWnd,
    )
```

hChart is the chart handle. *hWnd* is the window handle.

Xrt3dCallAction()

Calls an action explicitly at a given window coordinate. All coordinates should be within the graph area of the control. Any notification messages normally triggered by this action are called.

```
void
Xrt3dCallAction(
    HXRT3D      hChart,
    Xrt3dAction action,
    int         x,
    int         y
    )
```

hChart is the chart handle. *action* is the action to be called. *x* and *y* specify the window coordinate.

Xrt3dComputePalette()

Asks the chart to recompute its palette based on the current system palette.

```
void
Xrt3dComputePalette(
    HXRT3D      hChart,
    )
```

hChart is the chart handle.

Xrt3dComputeZValue()

Procedure which is useful when interactively dragging the value of a grid location to a new value.

```
double
Xrt3dComputeZValue(
    HXRT3D       hChart,
    int          xindex, yindex,
    int          pixel_x, pixel_y
    )
```

hChart is the chart handle; *xindex* and *yindex* identify the grid cell that is to be changed; *pixel_x* and *pixel_y* point to the new location of the screen to which the point has been dragged. If the grid value at (*xindex*, *yindex*) is a hole, then **Xrt3dComputeZValue**() returns the hole value for the grid.

Xrt3dContourStylesFromFile()

Procedure which creates a **NULL**-terminated array of pointers to **Xrt3dContourStyle** structures using information read in from a text file. The result can be used for setting the **XRT3D_CONTOUR_STYLES** property. File names of files that can be read with this routine may be used to set the **XRT3D_CONTOUR_STYLES** property in property files.

```
Xrt3dContourStyle **
Xrt3dContourStylesFromFile(
    char    *filename,
    char    *errbuf
    )
```

filename is the filename of the data file. If **Xrt3dContourStylesFromFile**() encounters any errors, it will return **NULL**, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to *stderr*. *errbuf* should be at least 100 bytes.

Lines in the file beginning with a "!" symbol in the first position are treated as comments and ignored. All other lines define one contour style, and should be of the form:

```
fill-color line-color line-width line-pattern
```

An example file:

```
! Example contour styles file
! This file defines 3 contour styles
#001e001effdc black 1 solid
#001e08d4ffdc red 2 dotted
#001e118affdc yellow 1 shortdash
```

Xrt3dContourStylesToFile()

Procedure which writes **Xrt3dContourStyle** structures to a text file. Returns a non-zero value if the operation is successful, and 0 if an error occurs.

```
int
Xrt3dContourStylesToFile(
    Xrt3dContourStyle   **styles,
    char                *filename,
    char                *errbuf
    )
```

styles is a pointer to the array of contour styles to write. *filename* is the pathname of the data file. If **Xrt3dContourStylesToFile**() encounters any errors, it will return 0, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to *stderr*. *errbuf* should be at least 100 bytes.

Xrt3dCreate()

Creates a chart without creating a window for it. Returns the chart handle.

```
HXRT3D
Xrt3dCreate(void)
```

Xrt3dCreateWindow()

Creates a window and a chart and attaches the two. Returns the chart handle.

```
HXRT3D
Xrt3dCreateWindow(
    LPCTSTR     lpWindowName,
    int         x, y,
    int         nWidth, nHeight,
    HWND        hWndParent,
    HINSTANCE   hInstance
    )
```

lpWindowName is a pointer to the window name. *x* and *y* are the horizontal and vertical position of the window. *nWidth* and *nHeight* are the width and height of the window. *hWndParent* is a handle to the parent or owner window. *hInstance* is a handle to the application instance.

Xrt3dDataCopy())

Creates a copy of the supplied data set.

```
Xrt3dData *
Xrt3dDataCopy(
    Xrt3dData *source
    )
```

Xrt3dDataShaded()

Creates a new **Xrt3dData** structure from an existing structure, filling it with a shaded-relief map of the original dataset, suitable for draping over the original data as a 4D dataset.

```
Xrt3dData *
Xrt3dDataShaded(
    Xrt3dData   *source,
    double      sweep,
    double      rise,
    double      scale,
    double      ambient,
    double      intensity
    )
```

This function calculates reflectance values for each of the facets in the original surface and stores the values in a new grid. This grid, when viewed with a set of grey-scale contour styles, resembles a shaded-relief map of the original data. The light source is located by the two angles *sweep* and *rise*. The difference in units between the "ground" coordinates in the XY plane and those of the Z values can be controlled through the *scale* parameter. If *scale* is 0, a default value of *scale* will be computed based on the average slope of the facets. The base amount of light for the surface is called the *ambient* light, while the brightness of the light is controlled through the *intensity*; both of these values should be between 0 and 1.

Xrt3dDataSmooth()

Passes a center-weighted averaging function over the grid, resulting in a smoother-looking dataset.

```
void
Xrt3dDataSmooth(
    Xrt3dData   *source,
    double      center_weight,
    )
```

center_weight must be between 0 and 1. This procedure is a nine-point filter using all points adjacent to the center. Each new value is a weighted average of the original value and its surrounding points. The weights of the surrounding points are computed based on the value of *center_weight* and its distance from the center point.

Xrt3dDataWindow()

Creates a new **Xrt3dData** structure from an existing structure, extracting a subset of the original, and resampling the data to the requested specifications.

```
Xrt3dData *
Xrt3dDataWindow(
    Xrt3dData           *source,
    double              xstart, ystart,
    double              xend, yend,
    int                 numx, numy,
    Xrt3dInterpMethod   interp
    )
```

The source data can be either a regular or irregular grid. The new dataset is a regularly-gridded dataset, with its origin at (*xstart*, *ystart*), *numx* lines in the *x*-direction, and *numy* lines in the *y*-direction in a region ending at (*xend*, *yend*). Any

requested portion that is outside of the region defined by the source grid is filled with holes. *interp* specifies either cubic (**XRT3D_INTERP_CUBIC_SPLINE**) or linear (**XRT3D_INTERP_LINEAR_SPLINE**) spline interpolation.

Xrt3dDestroyData()

Destroys an **Xrt3dData** structure.

```
void
Xrt3dDestroyData(
    Xrt3dData   *data,
    BOOL        all
    )
```

data is the structure to be destroyed. If *all* is **TRUE**, then any space allocated for data values is also freed.

This data structure is detached from any control that is currently using it.

Xrt3dDetachWindow()

Detaches a chart from its window. Returns the window handle.

```
HWND
Xrt3dDetachWindow(
    HXRT3D      hChart
    )
```

hChart is the chart handle.

Xrt3dDistnIndex()

Calculates the index of the entry in the distribution table for this value. The returned value can be used to access the contour style used for *value* by calling **Xrt3dGetNthContourStyle**().

```
int
Xrt3dDistnIndex(
    HXRT3D      hChart,
    double      value
    )
```

hChart is the chart handle.

Xrt3dDistnTableFromFile()

Procedure which creates a pointer to **Xrt3dDistnTable** structures using information read in from a text file. The result can be used for setting the **XRT3D_DISTN_TABLE** property. File names of files that can be read with this routine may be used to set the **XRT3D_DISTN_TABLE** property in property files.

```
Xrt3dDistnTable *
Xrt3dDistnTableFromFile(
    char    *filename,
    char    *errbuf
    )
```

filename is the filename of the data file. If **Xrt3dDistnTableFromFile**() encounters any errors, it will return **NULL**, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to *stderr*. *errbuf* should be at least 100 bytes.

Lines in the file beginning with a "!" symbol in the first position are treated as comments and ignored. All other lines define one distribution table entry.

An example file:

```
! Example distribution table file
! Note the not-quite-linear distribution!
1.0
1.2
1.4
3.0
```

Xrt3dDistnTableToFile()

Procedure which writes **Xrt3dDistnTable** structures to a text file. Returns a non-zero value if the operation is successful, and 0 if an error occurs.

```
int
Xrt3dDistnTableToFile(
    Xrt3dDistnTable *table,
    char            *filename,
    char            *errbuf
    )
```

table is a pointer to the structures to be written. *filename* is the pathname of the data file. If **Xrt3dDistnTableToFile**() encounters any errors, it will return 0, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to *stderr*. *errbuf* should be at least 100 bytes.

Xrt3dDrawToClipboard()

Copies a chart to the Windows clipboard as a bitmap or metafile.

```
BOOL
Xrt3dDrawToClipboard(
    HXRT3D          hChart,
    Xrt3dDrawFormat format
    )
```

hChart is the chart control to output. *format* is the graphics format to use and can be: **XRT3D_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT3D_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT3D_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF).

Xrt3dDrawToDC()

Outputs a chart to any device context as a bitmap or metafile.

```
BOOL
Xrt3dDrawToDC(
    HXRT3D          hChart,
    HDC             hdc,
    Xrt3dDrawFormat format,
    Xrt3dDrawScale  scale,
    long            left, top, width, height
    )
```

hChart is the chart control to output. *hdc* is a device context handle. *format* is the graphics format to use and can be: **XRT3D_DRAW_BITMAP** for a device-

independent bitmap (BMP); **XRT3D_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT3D_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF). *scale* specifies the scaling to perform when printing, and is one of **XRT3D_DRAWSCALE_NONE** (no scaling), **XRT3D_DRAWSCALE_TOWIDTH** (scale to width specified by *width*, preserving aspect ratio and ignoring *height*), **XRT3D_DRAWSCALE_TOHEIGHT** (scale to height specified by *height*, preserving aspect ratio and ignoring *width*), **XRT3D_DRAWSCALE_TOFIT** (scale to minimum of *height* or *width*, preserving aspect ratio), **XRT3D_DRAWSCALE_TOMAX** (enlarge to size of page regardless of aspect ratio). *left* is the offset from the left of the page. *top* is the offset from the top of the page. *width* specifies the width to scale to; to use the existing window width, set *width* to 0. *height* specifies the height to scale to; to use the existing window height, set *height* to 0.

Xrt3dDrawToFile()

Outputs a chart to a file as a bitmap or metafile.

```
BOOL
Xrt3dDrawToFile(
    HXRT3D          hChart,
    char            *filename,
    Xrt3dDrawFormat format
    )
```

hChart is the chart control to output. *format* is the graphics format to use and can be: **XRT3D_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT3D_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT3D_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF).

Xrt3dDupContourStyles()

Duplicates an array of **Xrt3dContourStyle** structures. The pointer to the duplicate array is returned.

```
Xrt3dContourStyle **
Xrt3dDupContourStyles(
    Xrt3dContourStyle **cs
    )
```

Xrt3dDupDistnTable()

Duplicates a **Xrt3dDistnTable** structure. The pointer to the duplicate table is returned. **NULL** is returned if the allocation failed for any reason.

```
Xrt3dDistnTable *
Xrt3dDupDistnTable(
    Xrt3dDistnTable *dt
    )
```

Xrt3dDupStrings()

Duplicates an array of strings. The pointer to the duplicate array is returned. **NULL** is returned if the allocation failed for any reason.

```
char **
Xrt3dDupStrings(
    char    **s
    )
```

Xrt3dDupValueLabels()

Duplicates a list of value labels.

```
Xrt3dValueLabel **
Xrt3dDupValueLabels(
    Xrt3dValueLabel **vlabels
    )
```

vlabels is an array of **NULL**-terminated pointers to structures of type **Xrt3dValueLabel**. A copy of this list is returned to the caller. **NULL** is returned if the allocation failed for any reason.

Xrt3dDupXYColors()

Duplicates a list of XYColors.

```
Xrt3dXYColor **
Xrt3dDupXYColors(
    Xrt3dXYColor **xycols
    )
```

xycols is an array of **NULL**-terminated pointers to structures of type **Xrt3dXYColor**. A copy of this list is returned to the caller. **NULL** is returned if the allocation failed for any reason.

Xrt3dFreeContourStyles()

Frees the memory used by an array of **Xrt3dContourStyle** structures.

```
void
Xrt3dFreeContourStyles(
    Xrt3dContourStyle **cs
    )
```

Xrt3dFreeDistnTable()

Frees the memory used by an **Xrt3dDistnTable** structure.

```
void
Xrt3dFreeDistnTable(
    Xrt3dDistnTable *dt
    )
```

Xrt3dFreePropString()

Frees the memory used by a string allocated by **Xrt3dGetPropString**().

```
void
Xrt3dFreePropString(
    char    *str
    )
```

Xrt3dFreeStrings()

Frees the memory used by an array of strings.

```
void
Xrt3dFreeStrings(
    char    **s
    )
```

Xrt3dFreeValueLabels()

Frees a list of value labels.

```
void
Xrt3dFreeValueLabels(
    Xrt3dValueLabel **vlabels
    )
```

vlabels is an array of **NULL**-terminated pointers to structures of type **Xrt3dValueLabel**. The memory occupied by these structures, and the pointer array, is freed.

Never free an **Xrt3dValueLabel** list that is currently being used by the control.

Xrt3dFreeXYColors()

Frees a list of XYColors.

```
void
Xrt3dFreeXYColors(xycols)
    Xrt3dXYColors **xycols;
```

xycols is an array of **NULL**-terminated pointers to structures of type **Xrt3dXYColor**. The memory occupied by these structures, and the pointer array, is freed.

Never free an **Xrt3dXYColor** list that is currently being used by the control.

Xrt3dGetAction()

Returns the action that is bound to this window event. If there is no action bound, **XRT3D_ACTION_NONE** is returned.

```
Xrt3dAction
Xrt3dGetAction(
    HXRT3D      hChart,
    UINT        msg,
    UINT        modifier,
    UINT        keycode
    )
```

hChart is the chart handle. *msg* is the message for this window event. The following messages are recognized:

WM_LBUTTONDBLCLK double-click left mouse button

WM_MBUTTONDBLCLK double-click both mouse buttons

WM_RBUTTONDBLCLK double-click right mouse button

WM_LBUTTONDOWN press left mouse button

WM_MBUTTONDOWN press both mouse buttons

WM_RBUTTONDOWN	press right mouse button
WM_LBUTTONUP	release left mouse button
WM_MBUTTONUP	release both mouse buttons
WM_RBUTTONUP	release right mouse button
WM_MOUSEMOVE	move mouse
WM_KEYDOWN	press key
WM_KEYUP	release key

modifier specifies the modifier flags, if any. The following modifier flags are recognized:

MK_LBUTTON	left mouse button
MK_MBUTTON	both mouse buttons
MK_RBUTTON	right mouse button
MK_ALT	Alt key
MK_SHIFT	Shift key
MK_CONTROL	Ctrl key

All actions are normalized to match the event sent by Microsoft Windows. For example, **MK_LBUTTON** is added to the modifier flags if a **WM_LBUTTONDOWN** message is sent.

keycode is the keycode. Any valid **VK_** value is treated as a recognized keycode. All alphabetic characters are forced to upper case. **MK_SHIFT** must appear in the modifier if capitals are desired. The CapsLock key toggles the meaning of the **MK_SHIFT** modifier.

Xrt3dGetActionList()

Returns a pointer to the entire list of actions used by the control. The list is stored as a linked list, and must not be modified in any way.

```
Xrt3dActionItem *
Xrt3dGetActionList(
    HXRT3D      hChart,
    )
```

hChart is the chart handle.

Xrt3dGetDistnTable()

Returns a pointer to the distribution table currently being used by the control.

```
Xrt3dDistnTable *
Xrt3dGetDistnTable(
    HXRT3D  graph
    )
```

This procedure allocates space for the returned table, which may be freed by the application by calling **Xrt3dFreeDistnTable**(). Returns **NULL** on failure for any reason.

Xrt3dGetHandle()

Retrieves the chart attached to a window.

```
HXRT3D
Xrt3dGetHandle(
    HWND        hWnd
    )
```

hWnd is the window handle.

Xrt3dGetNthContourStyle()

Returns a pointer to a contour style.

```
Xrt3dContourStyle *
Xrt3dGetNthContourStyle(
    HXRT3D      hChart,
    int         index,
    BOOL        used
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which contour style should be returned. Remember that the subset of contour styles actually used will depend on the number of levels (*nlevels*) in the distribution. Set *used* to **TRUE** to lookup contour styles from the set of contour styles actually being used. Set *used* to **FALSE** to lookup contour styles from the set of all contour styles currently defined. The pointer returned by this method should be considered read-only.

Xrt3dGetNthDataLabel()

Returns a pointer to a data label.

```
char *
Xrt3dGetNthDataLabel(
    HXRT3D      hChart,
    Xrt3dAxis   axis,
    int         index
    )
```

hChart is the chart handle. *axis* identifies the axis the label is to be taken from. *index* is the 0-indexed number of the requested label. If there are no data labels for this axis, or if *index* is out of range, **Xrt3dGetNthDataLabel**() returns **NULL**.

Xrt3dGetNthFooterString()

Returns a pointer to the *n*th footer string currently being used by the chart.

```
char *
Xrt3dGetNthFooterString(
    HXRT3D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which footer string should be returned. The default footer string for *index* will be returned if *index* is larger than the number of currently defined footer strings. The pointer returned by this method should be considered read-only.

Xrt3dGetNthHeaderString()

Returns a pointer to the *n*th header string currently being used by the chart.

```
char *
Xrt3dGetNthHeaderString(
    HXRT3D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which header string should be returned. The default header string for *index* will be returned if *index* is larger than the number of currently defined header strings. The pointer returned by this method should be considered read-only.

Xrt3dGetNthLegendString()

Returns a pointer to the *n*th legend string currently being used by the chart.

```
char *
Xrt3dGetNthLegendString(
    HXRT3D      hChart,
    int         index
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which legend string should be returned. The default legend string for *index* will be returned if *index* is larger than the number of currently defined legend strings. The pointer returned by this method should be considered read-only.

Xrt3dGetPalette()

Retrieves a chart's color palette.

```
HPALETTE
Xrt3dGetPalette(
    HXRT3D      hChart
    )
```

hChart is the chart handle.

Xrt3dGetPropString()

Retrieves the current value of a chart property as a string.

```
BOOL
Xrt3dGetPropString(
    HXRT3D      hChart,
    int         property,
    char        **str          /* Returned */
    )
```

hChart is the chart handle. *property* specifies any Olectra Chart property; *str* is a pointer to a string returned with the value of the property. *str* must be freed after use by calling **Xrt3dFreePropString**().

Xrt3dGetValueLabel()

Returns a pointer to the value label which has the closest value to the supplied value label.

```
Xrt3dValueLabel *
Xrt3dGetValueLabel(
    HXRT3D           hChart,
    Xrt3dAxis        axis,
    Xrt3dValueLabel  *vlabel
    )
```

hChart is the chart handle. *axis* identifies the axis the label is to be taken from. *vlabel* points to an **Xrt3dValueLabel** structure which has the *value* element filled in. A pointer to the value label which lies closest to the requested value is returned. If there are no value labels for this *axis*, **Xrt3dGetValueLabel**() returns **NULL**.

Xrt3dGetValues()

Retrieves the current value of one or more chart properties.

```
void
Xrt3dGetValues(
    HXRT3D     hChart,
    ...,
    NULL
    )
```

hChart is the chart handle. ... is one or more *property-value* pairs. Each pair consists of a *property* name and a pointer to a variable for its *value*. Olectra Chart writes the current value of the property to this variable. The list is terminated by **NULL**.

Xrt3dGetWindow()

Retrieves the window attached to a chart.

```
HWND
Xrt3dGetWindow(
    HXRT3D     hChart
    )
```

hChart is the chart handle.

Xrt3dGetXYColor()

Returns a pointer to a color in a bar chart.

```
COLORREF
Xrt3dGetXYColor(
    HXRT3D  hChart,
    int     xindex, yindex
    )
```

hChart is the chart handle. *xindex* and *yindex* specify the bar or set of bars to retrieve the color from.

Xrt3dMakeDataFromFile()

Procedure which will allocate an **Xrt3dData** structure and load it with data read in from a text file. See also **Xrt3dSaveDataToFile**().

```
Xrt3dData *
Xrt3dMakeDataFromFile(
    char    *filename,
    char    *errbuf
    )
```

filename is the filename of the data file. If **Xrt3dMakeDataFromFile**() encounters any errors, it will return **NULL**, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to *stderr*. *errbuf* should be at least 100 bytes.

Lines in the file beginning with a "!" symbol in the first position are treated as comments and ignored. The first non-comment line must begin with the data type, which may be **GRID** or **IRGRID**. This is followed by two integers (which should be ≥ 1) specifying the number of points in the X and Y directions. The next value is a floating-point number specifying the data hole value.

If **GRID** was specified, the next two floating-point numbers specify the grid step amount in the X and Y direction. The grid step amounts are followed by two floating-point numbers which indicate the X and Y grid origin values. Finally, the surface data values are listed in order of increasing Y values. (All of the Y values for X=origin are listed. Then all the Y values for X = (x origin + x grid step) are listed, etc.)

An example of a suitable **GRID** data file:

```
! Grid has 50 by 30 points
! Holes have value 100.0
! Grid increases in
!       X steps of 1.0 & Y steps of 2.0
! Origin of grid is at X = -20.0, Y = 50.0
GRID 50 30
100.0 1.0 2.0 -20.0 50.0
! 1500 data values follow, one for each grid pt.
49.875000 43.765625 38.500000 33.984375 30.12400
26.828125 24.000000 21.546875 19.375000 17.39062
. . .
```

If **IRGRID** is specified, then a list of xgrid and ygrid values is supplied instead of the origin, xstep and ystep values.

An example of a suitable **IRGRID** data file:

```
! Irregular grid has 10 by 5 points
! Holes have value 100.0
! Ten x values are given
! Five y values are given
IRGRID 10 5
100.0
20 21.1 22.3 23 24.4 25.4 27.8 29.9 30.1 31.2
50.3 51.3 52.6 54.8 59.6
! 50 data values follow
23.34343 12.89239 11.99423 15.781212 18.81988
. . .
```

Xrt3dMakeGridData()

Procedure used to allocate an empty **Xrt3dData** structure for regularly gridded data. To allocate space for irregularly gridded data, use **Xrt3dMakeIrGridData**().

```
Xrt3dData *
Xrt3dMakeGridData(
    int     numx, numy,
    double  noval,
    double  xstep, ystep,
    double  xorig, yorig,
    BOOL    all
    )
```

numx and *numy* specify the number of elements in the grid in the X and Y directions and should always be ≥ 1. *noval* defines the value that Olectra Chart will treat as a data hole. *xstep* and *ystep* define the step between grid elements along the X and Y axis. *xorig* and *yorig* define the origin of the grid in axis coordinates. If *all* is **TRUE**, then space for the data values is also allocated. Otherwise, only the structure itself is allocated, and the passed values are assigned to the structure.

Xrt3dMakeIrGridData()

Procedure used to allocate an empty **Xrt3dData** structure for irregularly gridded data. To allocate space for regularly gridded data, use **Xrt3dMakeGridData**().

```
Xrt3dData *
Xrt3dMakeIrGridData(
    int         numx, numy,
    double      noval,
    BOOL        all
    )
```

numx and *numy* specify the number of elements in the grid in the X and Y directions and should always be ≥ 1. *noval* defines the value that Olectra Chart will treat as a data hole. If *all* is **TRUE**, then space for the data values, and the *xgrid* and *ygrid* arrays is also allocated. Otherwise, only the structure itself is allocated, and the passed values are assigned to the structure.

Xrt3dMap()

Maps a pixel coordinate to a chart coordinate.

```
Xrt3dRegion
Xrt3dMap(
    HXRT3D          hChart,
    int             pix_x, pix_y,
    Xrt3dMapResult  *map
    )
```

Used by an application to determine the chart coordinates corresponding to *pix_x* and *pix_y* (for the chart whose handle is specified by *hChart*). Results are returned in *map*. Typically used in an event handling procedure.

Xrt3dPick()

Picks the displayed data that is closest to the given pixel coordinate.

```
Xrt3dRegion
Xrt3dPick(
    HXRT3D              hChart,
    int                 pix_x, pix_y,
    Xrt3dPickResult     *pick
    )
```

Used by an application to determine what data (in terms of grid point indices) is displayed closest to the pixel at (*pix_x*, *pix_y*) for the chart whose handle is specified by *hChart*. Results are returned in *pick*. This is typically used in an event handling procedure.

Xrt3dPrint()

Outputs a chart to a printer using the Windows Print dialog box.

```
BOOL
Xrt3dPrint(
    HXRT3D          hChart,
    Xrt3dDrawFormat format,
    Xrt3dDrawScale  scale,
    long            left, top, width, height
    )
```

hChart is the chart control to output. *format* is the graphics format to use and can be: **XRT3D_DRAW_BITMAP** for a Windows bitmap (BMP); **XRT3D_DRAW_METAFILE** for a Windows metafile (WMF); or **XRT3D_DRAW_ENHMETAFILE** (***Windows 95*** and ***Windows NT*** applications only) for an enhanced metafile (EMF). *scale* specifies the scaling to perform when printing, and is one of **XRT3D_DRAWSCALE_NONE** (no scaling), **XRT3D_DRAWSCALE_TOWIDTH** (scale to width specified by *width*, preserving aspect ratio and ignoring *height*), **XRT3D_DRAWSCALE_TOHEIGHT** (scale to height specified by *height*, preserving aspect ratio and ignoring *width*), **XRT3D_DRAWSCALE_TOFIT** (scale to minimum of *height* or *width*, preserving aspect ratio), **XRT3D_DRAWSCALE_TOMAX** (enlarge to size of page regardless of aspect ratio). *left* is the offset from the left of the page. *top* is the offset from the top of the page. *width* specifies the width to scale to; to use the existing window width, set *width* to 0. *height* specifies the height to scale to; to use the existing window height, set *height* to 0.

Xrt3dResetContourStyles()

Specifies that Olectra Chart is to return to using the default contour styles.

```
void
Xrt3dResetContourStyles(
    HXRT3D      hChart,
    )
```

hChart is the chart handle.

Xrt3dSaveDataToFile()

Procedure which writes out the data stored in an **Xrt3dData** structure to a text file, in a format suitable for use with **Xrt3dMakeDataFromFile**().

```
int
Xrt3dSaveDataToFile(
    Xrt3dData   *data,
    char        *filename,
    char        *errbuf
    )
```

data must point to a valid **Xrt3dData** structure. *filename* is the filename of the data file. If **Xrt3dMakeDataFromFile**() encounters any errors, it will return **NULL**, and will write an error message in *errbuf*. If *errbuf* is **NULL**, error messages will be written to *stderr*. *errbuf* should be at least 100 bytes. This procedure returns 1 on success and 0 on failure.

Xrt3dSetAction()

Programs an action for the Microsoft Windows event. Any previous action for this event is replaced.

```
void
Xrt3dSetAction(
    HXRT3D      hChart,
    UINT        msg,
    UINT        modifier,
    UINT        keycode,
    Xrt3dAction action
    )
```

hChart is the chart handle. *msg* is the message for this window event. *modifier* specifies the modifier flags, if any. *keycode* is the keycode. *action* is the new action for this event; if *action* is **XRT3D_ACTION_NONE**, the action mapping is completely removed.

Xrt3dSetNthContourStyle()

Sets the n^{th} contour style.

```
void
Xrt3dSetNthContourStyle(
    HXRT3D      hChart,
    int         index,
    Xrt3dContourStyle *cs,
    BOOL        used
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which contour style should be changed. *cs* points to an **Xrt3dContourStyle** structure containing the new contour style.

Remember that the subset of contour styles actually used will depend on the number of levels (*nlevels*) in the distribution. If *used* is **TRUE**, *index* refers only to the contour styles in use at this time. If *used* is **FALSE** *index* refers to the array of all contour styles defined.

Olectra Chart makes its own copy of the contour style pointed to by *cs*.

Xrt3dSetNthDataLabel()

Sets the n^{th} data label.

```
void
Xrt3dSetNthDataLabel(
    HXRT3D      hChart,
    Xrt3dAxis   axis,
    int         index,
    char        *label
    )
```

hChart is the chart handle. *axis* identifies the axis to be labelled. *index* is the 0-indexed number of the new label. *label* is the value of the new label. If *index* is greater than the number of labels in use, all intervening labels are assigned an empty string.

Xrt3dSetNthFooterString()

Sets the n^{th} footer string.

```
void
Xrt3dSetNthFooterString(
    HXRT3D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which footer string should be changed. *s* is the new string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

Xrt3dSetNthHeaderString()

Sets the n^{th} header string.

```
void
Xrt3dSetNthHeaderString(
    HXRT3D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which header string should be changed. *s* is the new string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

Xrt3dSetNthLegendString()

Sets the n^{th} legend string.

```
void
Xrt3dSetNthLegendString(
    HXRT3D      hChart,
    int         index,
    char        *s
    )
```

hChart is the chart handle. The (zero-based) *index* specifies which legend string should be changed. *s* is the new string to be used. Olectra Chart makes its own copy of the characters pointed to by *s*.

Xrt3dSetPropString()

Sets a chart property to the value represented by a string.

```
BOOL
Xrt3dSetPropString(
    HXRT3D      hChart,
    int         property,
    char        *str
    )
```

hChart is the chart handle. *property* specifies any Olectra Chart property; *str* is a pointer to a string representation to set the property to.

Xrt3dSetValueLabel()

Defines, replaces or deletes a value label.

```
void
Xrt3dSetValueLabel(
    HXRT3D            hChart,
    Xrt3dAxis         axis,
    Xrt3dValueLabel   *vlabel
    )
```

hChart is the chart handle. *axis* identifies the axis to be labelled. *vlabel* points to an **Xrt3dValueLabel** structure which has the *value* and *label* elements filled in. If *value* is the same as an existing value label for this axis, the new label replaces the existing label; if the new label is **NULL**, then the existing value label is deleted.

Due to round-off error locate the value label to be deleted or modified using **Xrt3dGetValueLabel**().

Xrt3dSetValues()

Sets one or more chart properties.

```
void
Xrt3dSetValues(
    HXRT3D      hChart,
    ...,
    NULL
    )
```

hChart is the chart handle. ... is one or more *property-value* pairs. Each pair consists of a *property* name and a variable (must be the same data type as the property) containing its new *value*. Olectra Chart sets the property to the value. The list is terminated by **NULL**.

Xrt3dSetXYColor()

Defines, replaces or deletes a color in a bar chart.

```
void
Xrt3dSetXYColor(
    HXRT3D        hChart,
    int           xindex, yindex,
    char          *color
    )
```

hChart is the chart handle. *xindex* and *yindex* specify the bar or set of bars to which the *color* will be applied. If *xindex* is -1, *color* is applied to all bars which are along the column defined by *yindex*; if *yindex* is -1, *color* is applied to all bars which are along the row defined by *xindex*; if both *xindex* and *yindex* are -1, the *color* is applied to all the bars in the chart.

Color requests are applied in the order they are specified. For instance, a request to assign the color blue to (-1, 3), followed by a request to assign the color red to (2, 3) results in bars in column 3 being blue, except the bar at *xindex* 2, which is red.

Olectra Chart will not keep more than one entry per (*xindex*, *yindex*) in its internal list of XYColors. If **Xrt3dSetXYColor**() is called with a color of **NULL**, the matching entry is removed from the list. Otherwise, any call to **Xrt3dSetXYColor**() places the new entry at the end of the list, and removes any duplicates from the list.

This procedure only has effect when **XRT3D_TYPE** is **XRT3D_TYPE_BAR**, and **XRT3D_DRAW_SHADED** is **TRUE**, and **XRT3D_DRAW_ZONES** is **FALSE**.

Xrt3dTextCreate()

Creates a text object. Returns its handle.

```
HXRT3DTEXT
Xrt3dTextCreate(void)
```

Xrt3dTextDestroy()

Destroys a text object.

```
void
Xrt3dTextDestroy(
    HXRT3DTEXT      text
    )
```

Xrt3dTextGetValues()

Retrieves the current value of one or more text object properties.

```
void
Xrt3dTextGetValues(
    HXRT3DTEXT  text,
    ...,
    NULL
    )
```

text is a text object handle. ... is one or more *property-value* pairs. Each pair consists of a *property* name and a pointer to a variable for its *value*. Olectra Chart writes the current value of the property to this variable. The list is terminated by **NULL**.

Xrt3dTextSetValues()

Sets one or more text object properties.

```
void
Xrt3dTextSetValues(
    HXRT3DTEXT  text,
    ...,
    NULL
    )
```

text is a text object handle. ... is one or more *property-value* pairs. Each pair consists of a *property* name and a variable (must be the same data type as the property) containing its new *value*. Olectra Chart sets the property to the value. The list is terminated by **NULL**.

Xrt3dUnmap()

Unmaps from a chart coordinate to a pixel coordinate.

```
void
Xrt3dUnmap(
    HXRT3D          hChart,
    double          x, y, z,
    Xrt3dMapResult  *map
    )
```

Used by an application to determine the pixel coordinates corresponding to the chart coordinate (*x*, *y*, *z*) in the chart whose handle is specified by *hChart*. Results are returned in the *pix_x* and *pix_y* elements of the structure pointed to by *map*.

pix_x and *pix_y* values of -1 are returned for unmap requests that are out of range.

Xrt3dUnpick()

Determines a pixel coordinate given a point on the surface grid.

```
void
Xrt3dUnpick(
    HXRT3D          hChart,
    int             xindex, yindex,
    Xrt3dPickResult *pick
    )
```

Used by an application to determine the pixel coordinates at which a data value is displayed (in the chart whose handle is specified by *hChart*). *xindex* and *yindex* are indices of the grid coordinate to be unpicked.

Results are returned in the *pix_x* and *pix_y* elements of the structure pointed to by *pick*. Both *pix_x* and *pix_y* will be -1 if the data is out of range.

Xrt3dZInterpolate()

Determines a Z-value of any point on the surface.

```
double
Xrt3dZInterpolate(
    HXRT3D  hChart,
    double  x, y
    )
```

This procedure returns an estimate of the surface value at (*x*, *y*) calculated using bilinear interpolation. *hChart* is the chart handle.

D

Message Reference

This appendix lists the Olectra Chart messages in alphabetical order.

XRT3DN_MODIFY_END

Sent to a chart's parent window to indicate that user interaction has ended:

```
XRT3DN_MODIFY_END:
hWnd = (HWND) wParam;
```

wParam is the window handle.

XRT3DN_MODIFY_START

Sent to a chart's parent window to indicate that a user interaction is about to begin:

```
XRT3DN_MODIFY_START:
hWnd = (HWND) wParam;
mcb = (Xrt3dModifyCallbackStruct *) lParam;

typedef struct {
    BOOL doit;
} Xrt3DModifyCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a modification message structure. In this structure, the *doit* element indicates whether the user interaction is to be permitted; set *doit* to **FALSE** to disallow this user interaction.

XRT3DN_PALETTECHANGED

Sent to a chart's parent window after the chart control has changed its color palette.

```
XRT3DN_PALETTECHANGED:
hWnd = (HWND) wParam;
```

wParam is the window handle.

XRT3DN_PROPERTIES

Sent to a chart's parent window to indicate that the user has requested to activate the property page:

```
XRT3DN_PROPERTIES:
hWnd = (HWND) wParam;
pcb = (Xrt3dPropertiesCallbackStruct *) lParam;

typedef struct {
    int x;
    int y;
} Xrt3dPropertiesCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a modification message structure. In this structure, the *x* and *y* elements indicate the coordinates at which the event occurred.

XRT3DN_REPAINTED

Sent to a window after the chart control has been redrawn:

```
XRT3DN_REPAINTED:
hWnd = (HWND) wParam;
cb = (Xrt3dCallbackStruct *) lParam;

typedef struct {
    HDC         hdc;
    RECT        rectDamaged;
} Xrt3dCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a repaint message structure. In this structure, *hdc* is the handle to the device context, and *rectDamaged* is the rectangle that has been repainted.

XRT3DN_RESIZED

Sent to a window after the chart control has changed size:

```
XRT3DN_RESIZED:
hWnd = (HWND) wParam;
rcb = (Xrt3dResizeCallbackStruct *) lParam;

typedef struct {
    int         width;
    int         height;
} Xrt3dResizeCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a resize message structure. In this structure, *width* is the new width of the control, and *height* is the new height of the control.

XRT3DN_ROTATE

Sent to a window when the user attempts to perform a rotation operation:

```
XRT3DN_ROTATE:
hWnd = (HWND) wParam;
rcb = (Xrt3dRotateCallbackStruct *) lParam;

typedef struct {
    double      xrotation;
    double      yrotation;
    double      zrotation;
    BOOL        doit;
} Xrt3dRotateCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a rotation message structure. In this structure, *xrotation*, *yrotation* and *zrotation* are the proposed values for the **XRT3D_[XYZ]ROTATION** properties; they can be modified by the message handler. The *doit* element indicates whether the rotation is to be permitted; set *doit* to **FALSE** to disallow this rotation.

XRT3DN_TRANSFORM

Sent to a window when the user attempts to perform a scaling, translation, or zooming operation:

```
XRT3DN_TRANSFORM:
hWnd = (HWND) wParam;
tcb = (Xrt3dTransformCallbackStruct *) lParam;

typedef struct {
    BOOL        reset;/* Read-only */
    double      scale;
    double      xtranslate;
    double      ytranslate;
    BOOL        doit;
} Xrt3dTransformCallbackStruct;
```

wParam is the window handle. *lParam* is a pointer to a transformation message structure. In this structure, *scale*, *xtranslate* and *ytranslate* are the proposed values for the **XRT3D_VIEW_SCALE**, **XRT3D_VIEW_XTRANSLATE** and **XRT3D_VIEW_YTRANSLATE** properties; they can be modified to limit the scope of the transformation. The *doit* element indicates whether the transformation is to be permitted; set *doit* to **FALSE** to disallow this transformation.

E

Data Types

This appendix lists the Olectra Chart data types in alphabetical order. The C language definition of structures is also provided.

Xrt3dAction

Enumeration used to specify Olectra Chart actions, used in user interaction.

```
XRT3D_ACTION_NONE
XRT3D_ACTION_MODIFY_START
XRT3D_ACTION_MODIFY_END
XRT3D_ACTION_MODIFY_CANCEL
XRT3D_ACTION_ROTATE
XRT3D_ACTION_SCALE
XRT3D_ACTION_TRANSLATE
XRT3D_ACTION_ZOOM_START
XRT3D_ACTION_ZOOM_UPDATE
XRT3D_ACTION_ZOOM_END
XRT3D_ACTION_ZOOM_CANCEL
XRT3D_ACTION_RESET
XRT3D_ACTION_PROPERTIES
XRT3D_ACTION_ROTATE_XAXIS
XRT3D_ACTION_ROTATE_YAXIS
XRT3D_ACTION_ROTATE_ZAXIS
XRT3D_ACTION_ROTATE_EYE
XRT3D_ACTION_ROTATE_FREE
```

Xrt3dActionItem

Structure used by **Xrt3dGetActionList()** to return the list of actions used by the control.

```
typedef struct tag_Xrt3dActionItem {
    UINT        msg;
    UINT        modifier;
    UINT        keycode;
    Xrt3dAction action;
    struct tag_Xrt3dActionItem *next;
} Xrt3dActionItem;
```

Xrt3dAdjust

Enumeration used by the **XRT3D_FOOTER_ADJUST** and **XRT3D_HEADER_ADJUST** properties:

```
XRT3D_ADJUST_LEFT
XRT3D_ADJUST_RIGHT
XRT3D_ADJUST_CENTER
```

Xrt3dAlign

Enumeration used with the **XRT3D_LEGEND_ORIENTATION** property:

```
XRT3D_ALIGN_VERTICAL
XRT3D_ALIGN_HORIZONTAL
```

Xrt3dAnchor

Enumeration used with the **XRT3D_LEGEND_ANCHOR** property:

```
XRT3D_ANCHOR_NORTH
XRT3D_ANCHOR_SOUTH
XRT3D_ANCHOR_EAST
XRT3D_ANCHOR_WEST
XRT3D_ANCHOR_NORTHWEST
XRT3D_ANCHOR_NORTHEAST
XRT3D_ANCHOR_SOUTHWEST
XRT3D_ANCHOR_SOUTHEAST
```

Xrt3dAnnoMethod

Enumeration used to specify the **XRT3D_[XYZ]ANNO_METHOD**:

```
XRT3D_ANNO_VALUES
XRT3D_ANNO_DATA_LABELS
XRT3D_ANNO_VALUE_LABELS
```

Xrt3dAxis

Enumeration used to specify an axis.

```
XRT3D_AXIS_X
XRT3D_AXIS_Y
XRT3D_AXIS_Z
XRT3D_AXIS_NONE
XRT3D_AXIS_EYE
```

Xrt3dBarFormat

Enumeration used to specify the **XRT3D_[XY]BAR_FORMAT**:

```
XRT3D_BAR_FIXED
XRT3D_BAR_HISTOGRAM
```

Xrt3dBorder

Enumeration used by the **XRT3D_HEADER_BORDER**, **XRT3D_FOOTER_BORDER**, **XRT3D_LEGEND_BORDER** and

XRT3D_GRAPH_BORDER properties, and the text object's **XRT3D_TEXT_BORDER** property:

```
XRT3D_BORDER_NONE
XRT3D_BORDER_3D_IN
XRT3D_BORDER_3D_OUT
XRT3D_BORDER_ETCHED_IN
XRT3D_BORDER_ETCHED_OUT
XRT3D_BORDER_SHADOW
XRT3D_BORDER_PLAIN
```

Xrt3dCallbackStruct

Structure that defines the information passed to the message handler when the chart control has been redrawn:

```
typedef struct {
    HDC             hdc;
    RECT            rectDamaged;
} Xrt3dCallbackStruct;
```

Xrt3dContourStyle

The structure defining how a particular contour and zone are displayed:

```
typedef struct {
    char            *fill_color;    /*for zoning*/
    char            *line_color;    /*line color*/
    int             line_width;     /*line width*/
    Xrt3dLinePattern lpat;          /*2D contours only*/
} Xrt3dContourStyle;
```

Xrt3dData

The structure defining data for use with **XRT3D_SURFACE_DATA** and with **XRT3D_ZONE_DATA**. It is a union of two data structures:

```
typedef union {
    Xrt3dGridData   g;
    Xrt3dIrGridData ig;
} Xrt3dData;
```

Xrt3dDataType

Enumeration used for defining the type of an **Xrt3dData** structure:

```
XRT3D_DATA_GRID
XRT3D_DATA_IRGRID
```

Xrt3dDistnMethod

Enumeration used for defining the distribution method used when DrawContours and/or DrawZones is **TRUE**. There are currently two methods:

```
XRT3D_DISTN_LINEAR
XRT3D_DISTN_FROM_TABLE
```

Xrt3dDistnTable

The structure used to define custom distributions:

```
typedef struct {
    int             nentries;
    double          *entry;
} Xrt3dDistnTable
```

Xrt3dDrawFormat

Enumeration used to specify the print format to use when printing the chart:

```
XRT3D_DRAW_BITMAP
XRT3D_DRAW_METAFILE
XRT3D_DRAW_ENHMETAFILE
```

Xrt3dDrawScale

Enumeration used to specify the scaling factor to use when printing the chart:

```
XRT3D_DRAWSCALE_NONE
XRT3D_DRAWSCALE_TOWIDTH
XRT3D_DRAWSCALE_TOHEIGHT
XRT3D_DRAWSCALE_TOFIT
XRT3D_DRAWSCALE_MAX
```

Xrt3dGridData

The structure defining **Xrt3dData** when the type is **XRT3D_DATA_GRID**:

```
typedef struct {
    Xrt3dDataType   type; /*XRT3D_DATA_GRID*/
    int             numx, numy;
    double          noval;
    double          xstep, ystep;
    double          xorig, yorig;
    double          **values;
} Xrt3dGridData;
```

Xrt3dInterpMethod

Enumeration used to define the interpolation method in **Xrt3dDataWindow**():

```
XRT3D_INTERP_LINEAR_SPLINE
XRT3D_INTERP_CUBIC_SPLINE
```

Xrt3dIrGridData

The structure defining **Xrt3dData** when the type is **XRT3D_DATA_IRGRID**:

```
typedef struct {
    Xrt3dDataType   type; /*XRT3D_DATA_IRGRID*/
    int             numx, numy;
    double          noval;
    double          *xgrid, *ygrid;
    double          **values;
} Xrt3dIrGridData;
```

Xrt3dLegendStyle

Enumeration of various legend styles, used with the **XRT3D_LEGEND_STYLE** property:

```
XRT3D_LEGEND_STYLE_STEPPED
XRT3D_LEGEND_STYLE_CONTINUOUS
```

Xrt3dLinePattern

Enumeration of various line patterns, used within an **Xrt3dContourStyle** structure:

```
XRT3D_LPAT_NONE
XRT3D_LPAT_SOLID
XRT3D_LPAT_LONG_DASH
XRT3D_LPAT_DOTTED
XRT3D_LPAT_SHORT_DASH
XRT3D_LPAT_LSL_DASH
XRT3D_LPAT_DASH_DOT
```

Xrt3dMapResult

Structure used to pass information about mapped pixel coordinates:

```
typedef struct {
    int         pix_x, pix_y;
    double      x, y, z;
} Xrt3dMapResult;
```

Xrt3dModifyCallbackStruct

Structure that defines the information passed to the message handler when a user interaction starts:

```
typedef struct {
    BOOL        doit;
} Xrt3dModifyCallbackStruct;
```

Xrt3dPickResult

Structure used to pass information about picked pixel coordinates:

```
typedef struct {
    int         pix_x, pix_y;
    int         xindex, yindex;
    int         distance;
} Xrt3dPickResult;
```

Xrt3dPreviewMethod

Enumeration used to specify how interactive rotations are to be presented to the user:

```
XRT3D_PREVIEW_CUBE
XRT3D_PREVIEW_FULL
```

Xrt3dPropertiesCallbackStruct

Structure that defines the information passed to the message handler when the user clicks the right mouse button to activate the property page:

```
typedef struct {
    int         x;
    int         y;
} Xrt3dPropertiesCallbackStruct;
```

Xrt3dRegion

Enumeration of map or pick results. Returned by **Xrt3dMap**() and **Xrt3dPick**():

```
XRT3D_RGN_NOWHERE
XRT3D_RGN_IN_GRAPH
XRT3D_RGN_IN_LEGEND
XRT3D_RGN_IN_FOOTER
XRT3D_RGN_IN_HEADER
```

Xrt3dResizeCallbackStruct

Structure that defines the information passed to the message handler when the chart control changes size:

```
typedef struct {
    int         width;
    int         height;
} Xrt3dResizeCallbackStruct;
```

Xrt3dRotateCallbackStruct

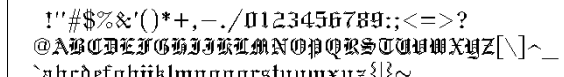
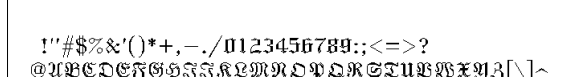
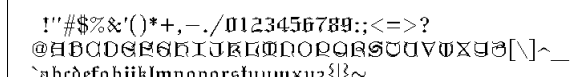
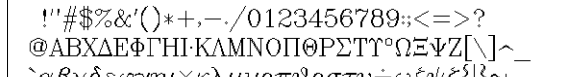
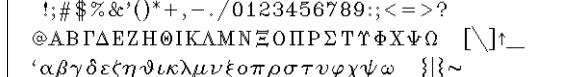
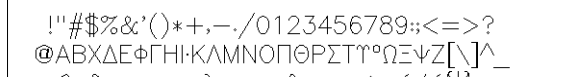
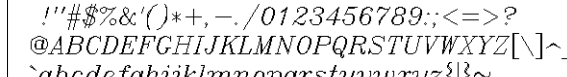
Structure that defines the information passed to the message handler when the user rotates a chart:

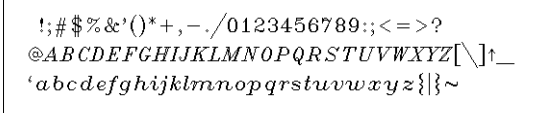
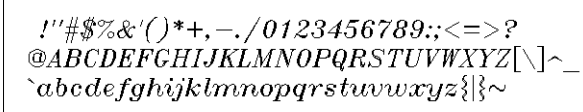
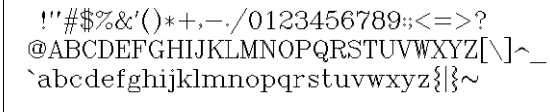
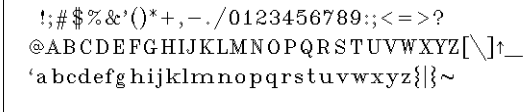
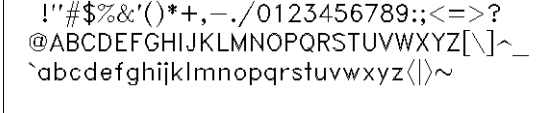
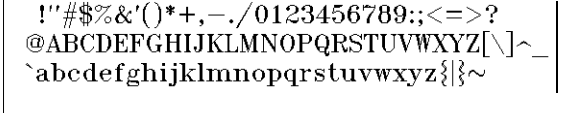
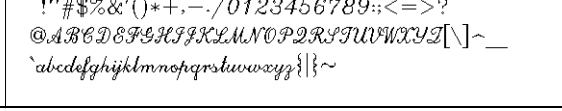
```
typedef struct {
    double      xrotation;
    double      yrotation;
    double      zrotation;
    BOOL        doit;
} Xrt3dRotateCallbackStruct;
```

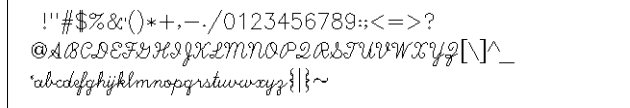
Xrt3dStrokeFont

Enumeration of stroke font names. Used by **XRT3D_AXIS_STROKE_FONT** and **XRT3D_AXIS_TITLE_STROKE_FONT**:

```
XRT3D_SF_CYRILLIC_COMPLEX
XRT3D_SF_GOTHIC_ENGLISH
XRT3D_SF_GOTHIC_GERMAN
XRT3D_SF_GOTHIC_ITALIAN
XRT3D_SF_GREEK_COMPLEX
XRT3D_SF_GREEK_COMPLEX_SMALL
XRT3D_SF_GREEK_SIMPLEX
XRT3D_SF_ITALIC_COMPLEX
XRT3D_SF_ITALIC_COMPLEX_SMALL
XRT3D_SF_ITALIC_TRIPLEX
XRT3D_SF_ROMAN_COMPLEX
XRT3D_SF_ROMAN_COMPLEX_SMALL
XRT3D_SF_ROMAN_DUPLEX
XRT3D_SF_ROMAN_SIMPLEX
XRT3D_SF_ROMAN_TRIPLEX
XRT3D_SF_SCRIPT_COMPLEX
XRT3D_SF_SCRIPT_SIMPLEX
```

Stroke Font Name	Glyphs
Cyrillic Complex	
Gothic English	
Gothic German	
Gothic Italian	
Greek Complex	
Greek Complex Small	
Greek Simplex	
Italic Complex	

Stroke Font Name	Glyphs
Italic Complex Small	
Italic Triplex	
Roman Complex	
Roman Complex Small	
Roman Duplex	
Roman Simplex	
Roman Triplex	
Script Complex	

Stroke Font Name	Glyphs
Script Simplex	

Xrt3dTextAttachMethod

Enumeration used to specify how Olectra Chart text objects are attached to the chart:

```
XRT3D_ATTACH_INDEX
XRT3D_ATTACH_PIXEL
XRT3D_ATTACH_POINT
```

Xrt3dTransformCallbackStruct

Structure that defines the information passed to the message handler when the user scales, translates or zooms a chart:

```
typedef struct {
    double      scale;
    double      xtranslate;
    double      ytranslate;
    BOOL        doit;
} Xrt3dTransformCallbackStruct;
```

Xrt3dType

Enumeration used to specify the **XRT3D_TYPE**:

```
XRT3D_TYPE_SURFACE
XRT3D_TYPE_BAR
```

Xrt3dValueLabel

Structure used to supply value labels for an axis:

```
typedef struct {
    double      value;
    char        *label;
} Xrt3dValueLabel;
```

Xrt3dXYColor

Structure used to specify coloring of individual bars.

```
typedef struct {
    int         xindex;
    int         yindex;
    COLORREF    *color;
} Xrt3dXYColor;
```

Xrt3dZoneMethod

Enumeration used to specify the zone filling method:

```
XRT3D_ZONE_CONTOURS
XRT3D_ZONE_CELLS
```

F

Sample Code

SIMPLE.C

This appendix provides an overview of the sample code included with Olectra Chart. There are two types of programming samples–examples and demos. Examples are short programs that illustrate *specific* features of the control. Demos are more complete applications that illustrate *several features* working together. Most common Olectra Chart programming tasks are covered in the examples and demos.

This appendix describes Olectra Chart's sample code and provides a listing of a program.

C Examples

Olectra Chart provides the following examples, located in Olectra Chart's \CHART\3D\DEMOS\DLL\SDK directory. There is a directory for each program.

- SIMPLE is the example program discussed in Chapter 1.
- PROFILE is a program that displays a two-dimensional profile of a three-dimensional surface.

MFC Example

Olectra Chart provides a simple program that uses the MFC classes. It is located in Olectra Chart's \CHART\3D\DEMOS\DLL\MFC directory.

OWL Example

Olectra Chart provides a simple program that uses the OWL classes. It is located in Olectra Chart's \CHART\3D\DEMOS\DLL\OWL directory.

F.1 SIMPLE.C

This program is the simple Olectra Chart program discussed in Chapter 1.

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <olch3d.h>

Xrt3dData* pData3d = NULL;

HANDLE ghInst;
HWND gMainHwnd;
HXRT3D hChart;
HBRUSH hbrushBack;
HANDLE hDrawMesh;
HANDLE hDrawShaded;
HANDLE hDrawContours;
HANDLE hDrawZones;

/* calculate the 3d data for the chart */
Xrt3dData *
CalculateGrid()
{
    int i, j;
    int nRows = 20;
    int nColumns = 20;
    double x, y;

    Xrt3dGridData *pGridData = (Xrt3dGridData *)
    Xrt3dMakeGridData(nRows, nColumns, XRT3D_HUGE_VAL,
        8.0 / (nRows - 1), 8.0 / (nColumns - 1),
        -3.0, -3.0, TRUE);

    if (!pGridData) {
        return (Xrt3dData*) NULL;
    }

    for (i = 0; i < nRows; i++) {
        x = pGridData->xorig + i * pGridData->xstep;
        for (j = 0; j < nColumns; j++) {
            y = pGridData->yorig + j * pGridData->ystep;
            pGridData->values[i][j] = 3*x*y - x*x*x - y*y*y;
        }
    }

    return (Xrt3dData*) pGridData;
}

/* Create all the controls used in the main windows */
void CreateControls(HWND parent)
{
    HDC hdc;
    TEXTMETRIC tm;
    int cxChar, cyChar;
    int controlWidth;
    COLORREF    crBack;

    hdc = GetDC(parent) ;
```

```
    SetMapMode(hdc, MM_TEXT);
    SelectObject(hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
    GetTextMetrics(hdc, &tm) ;
    cxChar = tm.tmAveCharWidth ;
    cyChar = tm.tmHeight + tm.tmExternalLeading ;

    controlWidth = cxChar * 16;
    ReleaseDC(parent, hdc) ;

    /*
     *  Create the 3d chart and their controls.
     *  The size and position of the chart will be recalculated
     *  during the WM_SIZE message.
     */

    hChart = Xrt3dCreateWindow("simple", 0, 0, 0, 0, parent, ghInst);
    hDrawMesh = CreateWindow("BUTTON", "DrawMesh",
        WS_VISIBLE | WS_CHILD | BS_AUTOCHECKBOX,
        cxChar, cyChar/2, controlWidth, cyChar,
        parent, NULL, ghInst, NULL);
    hDrawShaded = CreateWindow("BUTTON", "DrawShaded",
        WS_VISIBLE | WS_CHILD | BS_AUTOCHECKBOX,
        cxChar+4+controlWidth, cyChar/2, controlWidth, cyChar,
        parent, NULL, ghInst, NULL);
    hDrawContours = CreateWindow("BUTTON", "DrawContours",
        WS_VISIBLE | WS_CHILD | BS_AUTOCHECKBOX,
        cxChar+4+(controlWidth*2), cyChar/2, controlWidth, cyChar,
        parent, NULL, ghInst, NULL);
    hDrawZones = CreateWindow("BUTTON", "DrawZones",
        WS_VISIBLE | WS_CHILD | BS_AUTOCHECKBOX,
        cxChar+4+(controlWidth*3), cyChar/2, controlWidth, cyChar,
        parent, NULL, ghInst, NULL);

    /*
     * get the current background color used by the demo
     * and use that color for the background of the rest of the window
     */
    Xrt3dGetValues(hChart, XRT3D_BACKGROUND_COLOR, &crBack, NULL);
    hbrushBack = CreateSolidBrush(crBack);

#ifdef _WIN32
    SetClassLong(gMainHwnd, GCL_HBRBACKGROUND, (long) hbrushBack);
#else
    SetClassWord(gMainHwnd, GCW_HBRBACKGROUND, (WORD) hbrushBack);
#endif
}

/* Resize/reposition the chart based on the new size of the main window
*/
void SizeChart(HWND hwnd, int width, int height)
{
    HDC hdc;
    TEXTMETRIC tm;
    int cyChar;
    int chartHeight;
    int chartWidth;
    int controlHeight;

    hdc = GetDC(hwnd);
    SetMapMode(hdc, MM_TEXT);
```

```
    SelectObject(hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
    GetTextMetrics(hdc, &tm) ;
    ReleaseDC(hwnd, hdc);

    cyChar = tm.tmHeight + tm.tmExternalLeading ;
    controlHeight = cyChar * 2;
    chartWidth = width;
    chartHeight = height - (controlHeight * 4);

    /* place chart in bottom portion of window */
    SetWindowPos(Xrt3dGetWindow(hChart), HWND_BOTTOM, 0,
                (controlHeight * 2),
                chartWidth, chartHeight, SWP_SHOWWINDOW | SWP_NOZORDER);

    InvalidateRect(gMainHwnd, NULL, TRUE);
}

/* draw routine */
void DrawChart(HWND hwnd, HDC hdc)
{
    TEXTMETRIC tm;
    RECT rect;
    HBRUSH hbrush;
    int cxChar, cyChar;
    int chartWidth;
    int controlHeight;

    SetMapMode(hdc, MM_TEXT);
    SelectObject(hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
    GetTextMetrics(hdc, &tm) ;
    cxChar = tm.tmAveCharWidth ;
    cyChar = tm.tmHeight + tm.tmExternalLeading ;

    controlHeight = cyChar * 2;
    GetClientRect(hwnd, &rect);
    chartWidth = rect.right - rect.left;

    /* place controls in top portion of window */
    MoveToEx(hdc, 0, controlHeight, NULL);
    LineTo(hdc, chartWidth, controlHeight);
#ifdef _WIN32
    hbrush = CreateSolidBrush(GetSysColor(COLOR_BTNFACE));
#else
    hbrush = CreateSolidBrush(GetSysColor(COLOR_WINDOW));
#endif
    rect.left = 0;
    rect.top = 0;
    rect.right = chartWidth;
    rect.bottom = controlHeight;
    FillRect(hdc, &rect, hbrush);
    DeleteObject(hbrush);

    ShowWindow(hDrawMesh, SW_SHOW);
    ShowWindow(hDrawShaded, SW_SHOW);
    ShowWindow(hDrawContours, SW_SHOW);
    ShowWindow(hDrawZones, SW_SHOW);
}


/* Main window WndProc */
```

```
LRESULT CALLBACK MainWndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM
lParam)
{
   HWND hwndCtrl;
   HDC hdc;
   PAINTSTRUCT ps;
   int nCheck;

   switch(msg) {

     case WM_COMMAND:

       /* handle commands */
       switch(LOWORD(wParam)) {

           /* handle messages from check buttons */
         default:
#ifdef _WIN32
           hwndCtrl = (HWND)lParam;
#else
           hwndCtrl = (HWND)LOWORD(lParam);
#endif
           if (hwndCtrl == hDrawMesh) {
              nCheck = (int) SendMessage(hwndCtrl, BM_GETCHECK, 0, 0L);
               Xrt3dSetValues(hChart, XRT3D_DRAW_MESH, nCheck, NULL);
           }
           else if (hwndCtrl == hDrawShaded) {
              nCheck = (int) SendMessage(hwndCtrl, BM_GETCHECK, 0, 0L);
               Xrt3dSetValues(hChart, XRT3D_DRAW_SHADED, nCheck, NULL);
           }
           else if (hwndCtrl == hDrawContours) {
              nCheck = (int) SendMessage(hwndCtrl, BM_GETCHECK, 0, 0L);
               Xrt3dSetValues(hChart, XRT3D_DRAW_CONTOURS, nCheck,
               NULL);
           }
           else if (hwndCtrl == hDrawZones) {
              nCheck = (int) SendMessage(hwndCtrl, BM_GETCHECK, 0, 0L);
               Xrt3dSetValues(hChart, XRT3D_DRAW_ZONES, nCheck, NULL);
           }
           break;
       }
       return(0);

     case WM_SIZE:
       /* resize the chart */
       SizeChart(hwnd, (int) LOWORD(lParam), (int) HIWORD(lParam));
       return(0);

     case WM_PAINT:
       hdc = BeginPaint(hwnd, &ps);
       DrawChart(hwnd, hdc);
       EndPaint(hwnd, &ps);
       break;

       /*
        *  To properly handle colors the chart needs to be
        *  notified about any system palette changes.
        */
     case XRT3DN_PALETTECHANGED:
       SendMessage(Xrt3dGetWindow(hChart), WM_QUERYNEWPALETTE, 0, 0);
```

```
        break;

      case WM_QUERYNEWPALETTE:
        /* Make sure the window is at the top of the Z-order so that
           the background palettes can be realized before any other
           application gets a chance to realize a palette.
         */
        /* In Windows for Workgroups, this message can be received
           before the window is moved to the top of the Z-order
           causing the first WM_PALETTECHANGED message to be sent
           not to this window but the window which was "previously"
           at the top of the Z-order.
         */
        BringWindowToTop(hwnd);

        SendMessage(Xrt3dGetWindow(hChart), WM_QUERYNEWPALETTE, 0, 0);
        break;

      case WM_PALETTECHANGED:
        SendMessage(Xrt3dGetWindow(hChart), msg, wParam, lParam);
        break;

      case WM_DESTROY:
        PostQuitMessage(0);
        return(0);
    }

    return(DefWindowProc(hwnd, msg, wParam, lParam));
}


/* Entry point for main program */
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
    WNDCLASS    wc;
    MSG         msg;

    memset(&wc, 0, sizeof(wc));
    wc.lpfnWndProc = (WNDPROC) MainWndProc;
    wc.hInstance = hInstance;
    wc.hIcon = LoadIcon(hInstance, "simpleicon");
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = GetStockObject(BLACK_BRUSH);
    wc.lpszMenuName = (LPSTR) "Menu";
    wc.lpszClassName = (LPSTR) "Chart3D_Simple";

    if (!RegisterClass(&wc)) {
        MessageBox(NULL, "Cannot RegisterClass()",
                   "Err! - TEST", MB_OK | MB_ICONEXCLAMATION);
        return(FALSE);
    }

    ghInst = hInstance;

    /* create main window */
    gMainHwnd = CreateWindow("Chart3D_Simple",
        "Olectra Chart 3D - Simple",
        WS_OVERLAPPEDWINDOW, 0, 0, 700, 400, NULL, NULL, ghInst, NULL);
    if (!gMainHwnd) {
```

```
        return 0;
    }

    /* create all controls in main window */
    CreateControls(gMainHwnd);

    /* calculate and set the 3d data for the chart */
    pData3d = CalculateGrid();
    Xrt3dSetValues(hChart, XRT3D_SURFACE_DATA, pData3d, NULL);
    Xrt3dSetValues(hChart, XRT3D_XAXIS_TITLE, "X Axis", NULL);
    Xrt3dSetValues(hChart, XRT3D_YAXIS_TITLE, "Y Axis", NULL);
    Xrt3dSetValues(hChart, XRT3D_ZAXIS_TITLE, "Z Axis", NULL);
    SendMessage(hDrawMesh, BM_SETCHECK, 1, 0L);

    ShowWindow(gMainHwnd, nCmdShow);
    UpdateWindow(gMainHwnd);

    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    /* remember to free up memory */
    Xrt3dDestroyData(pData3d, TRUE);

    DeleteObject(hbrushBack);

    return(msg.wParam);
}
```

Index

15 chart types 24
4D charts 65, 66
  creating 65

A

action maps
  changing 58
action maps and messages 55
actions
  calling directly 60
  determining mappings 59
  disabling 60
  modifier flags 59
  programming 58
  programming mappings 60
  recognized keycodes 59
axis
  displaying 34
  font and size 34
  labelling 38
  minimum/maximum bounds 34
  scaling 38
  title 34
axis labels
  data labels method 39
  value labels method 39
  value method 39
axis properties, summary 21

B

background color
  Text Object 70
background colors 43
  chart 43
  data area 43
  footer 43
  header 43
  legend 43
  window 43
bar chart properties, summary 22
bar charts
  4D 66
  coloring 31
batching property updates 15
border types 42
borders 42
  Text Object 70

C

C++
  MFC header file 16
  OWL header file 16
C++ classes
  CChart3d 16
  CChart3dData 16
  TChart3d 16
  TChart3dData 16
callback structures
  Modify 129
  Properties 129
  Resize 130
  Rotate 130
  Transform 133
changing action maps 58
chart
  foreground color 43
  grid lines 39
  output 44
  perspective effect 35
  positioning of areas 41
chart data area
  background color 43
chart mesh
  colors 36
  filtering 36
  hidden lines 37
chart properties, summary 21
chart surface
  colors 37
  solid 37
chart types 24
  4D 65, 66
  contours 26
  contours and zones 28
  DrawContours 25
  DrawMesh 24
  DrawShaded 25
  DrawZones 25
  mesh 26
  mesh and contour 27
  mesh and shaded 27
  mesh and zones 27

mesh, contours, zones 28
mesh, shaded, contours 28
mesh, shaded, contours, zones 29
mesh, shaded, zones 28
shaded 26
shaded and contour 27
shaded and zones 27
shaded, contours, zones 28
zones 26
class name 79
colors
background 43
bar charts 31
chart mesh 36
chart surface 37
foreground 43
palette handling 43
palette notification message 44
specifying 43
Text Object 70
contour charts 26
contour styles
customizing 73, 74
line patterns 74
usage tips 75
contours 25
contours and zones charts 28
control
class name 79
include file 79
synopsis 79
controlling interactive rotate 58
controlling interactive scale, transform, zoom 57

D

data
application responsibility 49
changing 49
convenience procedures 51
example of setting a value 51
input from file 48
irregular grid 50
overview of Xrt3dData structure 47
real-time performance 49
regular grid 49
data area
background color 43
data display properties, summary 20
default user interactions 53
disabling all user interaction 56
disabling and disallowing interactions 60
Distributing Olectra Chart Applications 16
distribution table, customizing 70
double buffering 44
DrawContours property 25
DrawMesh property 24
DrawShaded property 25
DrawZones property 25
DrawZones with DrawShaded 29

E

example programs, discussion of 135

F

fonts
and Microsoft Windows 3.1 15
header, footer and legend 40
Text Object 70
footer
border 42
font 40
foreground color 43
positioning 41
text 40
footer area, definition of 11
foreground color
Text Object 70
foreground colors 43
chart 43
footer 43
header 43
legend 43
window 43

G

graph
border 42
positioning 41
graph area, definition of 11
graph areas
illustration of 12
positioning property summary 41
grid lines 39
Grid text objects 67
characteristics 67

H

header
border 42
font 40
foreground color 43
positioning 41
text 40
header area, definition of 11
header/footer properties, summary 22
help support, see Getting Started booklet
holes in data 48

I

include file, Olectra Chart 79
introduction to Olectra Chart 1
irregular grid data 50

L

legend
  4D charts 67
  anchor 35
  bar chart colors 31
  border 35, 42
  customizing contents 35
  explicit positioning 35
  font 40
  foreground color 43
  layout 34
  orientation 34
  positioning 35, 41
  style 35
legend area, definition of 11
legend properties, summary 23
line patterns 74

M

manual, overview of 2
mesh
  colors 36
  filtering 36
  hidden lines 37
mesh and contour charts 27
mesh and shaded charts 27
mesh and zones charts 27
mesh charts 26
mesh, contours, zones charts 28
mesh, shaded, contours charts 28
mesh, shaded, contours, zones charts 29
mesh, shaded, zones charts 28
MFC
  header file 16
MFC C++ demo 135
MFC classes
  CChart3d 16
  CChart3dData 16
Microsoft Windows 3.1 and fonts 15
miscellaneous properties, summary 23
Modify callback structure 129

N

nentries, Xrt3dDistnTable structure 71
nlevels 71, 73, 75
noval element of Xrt3dData structure 48

O

Olectra Chart
  basic terminology 12
  class name 79
  include file 79
  introduction 7
  introduction to 1
  overview of manual 2
Olectra Chart Text Object
  attachment methods 67
  characteristics of Grid types 67
  characteristics of Pixel type 68
  definition of 67
  intersecting multiple objects 68
  positioning 69
  property summary 24, 68
  removing 70
  types 67
OlectraChart3D, class name 79
outputting charts 44
OWL
  header file 16
OWL C++ demo 135
OWL classes
  TChart3d 16
  TChart3dData 16

P

palette notification message 44
perspective effect 35
Pixel text objects 67
  characteristics 68
pointer properties, specifying 14
positioning graph areas, strategies 42
printing charts 44
PROFILE example program 135
programming action mappings 60
programming actions 58
properties
  batching updates 15
  pointer 14
  retrieving values 12
  setting values 12
  USE_DEFAULT 13
Properties callback structure 129
property summary
  axis 21
  bar charts 22
  chart 21
  data display 20
  header/footer 22
  legend 23
  miscellaneous 23

R

real-time performance 49
regular grid data 49
Repaint message 64
resetting interactions 57
Resize callback structure 130
Resize message 64
resizing windows 63
Rotate callback structure 130
rotating, interactive 53

S

sample code 135
scale of axis 38
scaling, interactive 53
shaded and contour charts 27
shaded and zones charts 27
shaded charts 26
shaded, contours, zones charts 28
simple example 7
SIMPLE example program 135
SIMPLE.C 7
stroke fonts, listing of 130
support, see Getting Started booklet
surface
  colors 37
  solid 37
surface charts
  4D 65
  creating 4D 65

T

technical support, see Getting Started booklet
terminology of Olectra Chart 12
Text Object
  attachment methods 67
  characteristics of Grid types 67
  characteristics of Pixel type 68
  definition 67
  intersecting multiple objects 68
  positioning 69
  property summary 24, 68
  removing 70
  types of 67
Transform callback structure 133
translation, interactive 53
translations and actions
  disabling 56
  rotating 53
  scaling 53
  translation 53
  zoom 53
types of charts 24

U

USE_DEFAULT properties 13
user interaction
  controlling rotate 58
  controlling scale, translate, zoom 57
  customizing 55
  disabling all 56
  ending 58
  resetting 57
  rotating 53
  scaling 53
  starting 55
  three stages 55
  translation 53
  updating 56
  with chart data 61
  zooming 53
user interaction, default 53

W

Windows integration 11
windows, resizing 63

X

XRT3D_AXIS_STROKE_FONT 34, 79
XRT3D_AXIS_STROKE_SIZE 79
XRT3D_AXIS_TITLE_STROKE_SIZE 80
XRT3D_BACKGROUND_COLOR 43, 80
XRT3D_BORDER 80
XRT3D_BORDER_WIDTH 80
XRT3D_CONTOUR_STYLES 25, 80
XRT3D_DATA_AREA_BACKGROUND_COLOR 43, 81
XRT3D_DEBUG 81
XRT3D_DISTN_METHOD 25, 70, 81
XRT3D_DISTN_TABLE 25, 81, 103
XRT3D_DOUBLE_BUFFER 44, 81
XRT3D_DRAW_CONTOURS 81
XRT3D_DRAW_HIDDEN_LINES 25, 37, 82
XRT3D_DRAW_MESH 82
XRT3D_DRAW_SHADED 82
XRT3D_DRAW_ZONES 82
XRT3D_FOOTER_ADJUST 40, 83
XRT3D_FOOTER_BACKGROUND_COLOR 83
XRT3D_FOOTER_BORDER 83
XRT3D_FOOTER_BORDER_WIDTH 83
XRT3D_FOOTER_FONT 83
XRT3D_FOOTER_FOREGROUND_COLOR 83
XRT3D_FOOTER_HEIGHT 83
XRT3D_FOOTER_STRINGS 40, 83
XRT3D_FOOTER_WIDTH 83
XRT3D_FOOTER_X 83
XRT3D_FOOTER_X_USE_DEFAULT 84
XRT3D_FOOTER_Y 83
XRT3D_FOOTER_Y_USE_DEFAULT 84

XRT3D_FOREGROUND_COLOR 43, 84
XRT3D_GRAPH_BACKGROUND_COLOR 84
XRT3D_GRAPH_BORDER 84
XRT3D_GRAPH_BORDER_WIDTH 84
XRT3D_GRAPH_FOREGROUND_COLOR 84
XRT3D_GRAPH_HEIGHT 84
XRT3D_GRAPH_HEIGHT_USE_DEFAULT 84
XRT3D_GRAPH_WIDTH 84
XRT3D_GRAPH_WIDTH_USE_DEFAULT 84
XRT3D_GRAPH_X 85
XRT3D_GRAPH_X_USE_DEFAULT 85
XRT3D_GRAPH_Y 85
XRT3D_GRAPH_Y_USE_DEFAULT 85
XRT3D_HEADER_ADJUST 40, 85
XRT3D_HEADER_BACKGROUND_COLOR 85
XRT3D_HEADER_BORDER 85
XRT3D_HEADER_BORDER_WIDTH 85
XRT3D_HEADER_FONT 85
XRT3D_HEADER_FOREGROUND_COLOR 43, 85
XRT3D_HEADER_HEIGHT 86
XRT3D_HEADER_STRINGS 40, 85
XRT3D_HEADER_WIDTH 86
XRT3D_HEADER_X 86
XRT3D_HEADER_X_USE_DEFAULT 86
XRT3D_HEADER_Y 86
XRT3D_HEADER_Y_USE_DEFAULT 86
XRT3D_HEIGHT 86
XRT3D_LEGEND_ANCHOR 41, 86
XRT3D_LEGEND_BACKGROUND_COLOR 86
XRT3D_LEGEND_BORDER 86
XRT3D_LEGEND_BORDER_WIDTH 86
XRT3D_LEGEND_FONT 86
XRT3D_LEGEND_FOREGROUND_COLOR 87
XRT3D_LEGEND_HEIGHT 88
XRT3D_LEGEND_LABEL_FUNC 72, 87
XRT3D_LEGEND_ORIENTATION 34, 87
XRT3D_LEGEND_SHOW 34, 87
XRT3D_LEGEND_STRINGS 67, 72, 87
XRT3D_LEGEND_STYLE 35, 87
XRT3D_LEGEND_WIDTH 88
XRT3D_LEGEND_X 88
XRT3D_LEGEND_X_USE_DEFAULT 88
XRT3D_LEGEND_Y 88
XRT3D_LEGEND_Y_USE_DEFAULT 88
XRT3D_MESH_BOTTOM_COLOR 25, 36, 88
XRT3D_MESH_TOP_COLOR 25, 36, 88
XRT3D_NAME 88
XRT3D_NUM_DISTN_LEVELS 70, 88
XRT3D_PERSPECTIVE_DEPTH 35, 88
XRT3D_PREVIEW_METHOD 56, 88
XRT3D_PROJECT_ZMAX 32, 89
XRT3D_PROJECT_ZMIN 32, 89
XRT3D_REPAINT 15, 89
XRT3D_SOLID_SURFACE 37, 89
XRT3D_SURFACE_BOTTOM_COLOR 37, 89
XRT3D_SURFACE_DATA 14, 49, 89
XRT3D_SURFACE_TOP_COLOR 37, 89
XRT3D_TEXT_ADJUST 68, 95
XRT3D_TEXT_ATTACH_INDEX_X 95
XRT3D_TEXT_ATTACH_INDEX_Y 95
XRT3D_TEXT_ATTACH_METHOD 67, 95
XRT3D_TEXT_ATTACH_PIXEL_X 95
XRT3D_TEXT_ATTACH_PIXEL_Y 95
XRT3D_TEXT_ATTACH_POINT_X 96
XRT3D_TEXT_ATTACH_POINT_Y 96
XRT3D_TEXT_ATTACH_POINT_Z 96
XRT3D_TEXT_BACKGROUND 70
XRT3D_TEXT_BACKGROUND_COLOR 96
XRT3D_TEXT_BORDER 70, 96
XRT3D_TEXT_BORDER_WIDTH 70, 96
XRT3D_TEXT_FONT 70, 96
XRT3D_TEXT_FOREGROUND 70
XRT3D_TEXT_FOREGROUND_COLOR 96
XRT3D_TEXT_LINE_SHOW 69, 96
XRT3D_TEXT_OFFSET_X 97
XRT3D_TEXT_OFFSET_Y 97
XRT3D_TEXT_PLANE 69, 97
XRT3D_TEXT_SHOW 70, 97
XRT3D_TEXT_STRINGS 68, 97
XRT3D_TEXT_STROKE_FONT 70, 97
XRT3D_TEXT_STROKE_SIZE 70, 97
XRT3D_TYPE 29, 89
XRT3D_VIEW_NORMALIZED 89
XRT3D_VIEW_SCALE 90
XRT3D_VIEW_XTRANSLATE 90
XRT3D_VIEW_YTRANSLATE 90
XRT3D_WIDTH 90
XRT3D_XANNO_METHOD 39, 90
XRT3D_XAXIS_SHOW 34, 90
XRT3D_XAXIS_TITLE 90
XRT3D_XBAR_FORMAT 90
XRT3D_XBAR_SPACING 91
XRT3D_XDATA_LABELS 91
XRT3D_XGRID_LINES 39, 91
XRT3D_XMAX 34, 91
XRT3D_XMAX_USE_DEFAULT 34, 92
XRT3D_XMESH_FILTER 36, 92
XRT3D_XMESH_SHOW 92
XRT3D_XMIN 34, 92
XRT3D_XMIN_USE_DEFAULT 34, 92
XRT3D_XROTATION 57, 93
XRT3D_XSCALE 38, 93
XRT3D_XTITLE 34
XRT3D_XVALUE_LABELS 93
XRT3D_XY_COLORS 31, 93
XRT3D_YANNO_METHOD 39, 90
XRT3D_YAXIS_SHOW 34, 90
XRT3D_YAXIS_TITLE 90
XRT3D_YBAR_FORMAT 90
XRT3D_YBAR_SPACING 91
XRT3D_YDATA_LABELS 91
XRT3D_YGRID_LINES 39, 91
XRT3D_YMAX 34, 91
XRT3D_YMAX_USE_DEFAULT 34, 92
XRT3D_YMESH_FILTER 36, 92
XRT3D_YMESH_SHOW 92
XRT3D_YMIN 34, 92
XRT3D_YMIN_USE_DEFAULT 34, 92
XRT3D_YROTATION 57, 93

XRT3D_YSCALE 38, 93
XRT3D_YTITLE 34
XRT3D_YVALUE_LABELS 93
XRT3D_Z_ONE_DATA 14
XRT3D_ZANNO_METHOD 39, 90
XRT3D_ZAXIS_SHOW 34, 90
XRT3D_ZAXIS_TITLE 90
XRT3D_ZGRID_LINES 39, 91
XRT3D_ZMAX 34, 91
XRT3D_ZMAX_USE_DEFAULT 34, 92
XRT3D_ZMIN 34, 92
XRT3D_ZMIN_USE_DEFAULT 34, 92
XRT3D_ZONE_DATA 25, 93
  use of as 4D bar chart 66
  use of for 4D charts 65
XRT3D_ZONE_METHOD 33, 93
XRT3D_ZORIGIN 25, 29, 94
XRT3D_ZROTATION 57, 93
XRT3D_ZSCALE 38, 93
XRT3D_ZTITLE 34
XRT3D_ZVALUE_LABELS 93
Xrt3dAction 125
Xrt3dActionItem 125
Xrt3dAdjust 126
Xrt3dAlign 126
Xrt3dAnchor 126
Xrt3dAnnoMethod 39, 126
Xrt3dAttachWindow() 99
Xrt3dAxis 126
Xrt3dBarFormat 126
Xrt3dBorder 126
Xrt3dCallAction() 60, 99
Xrt3dCallbackStruct structure 127
Xrt3dComputePalette() 99
Xrt3dComputeZValue 63
Xrt3dComputeZValue() 100
Xrt3dContourStyle structure 73, 100, 101, 105, 109, 115, 127
  fill_color 73
  line_color 73
  line_width 73
  lpat 74
Xrt3dContourStylesFromFile() 100
Xrt3dContourStylesToFile() 101
Xrt3dCreate() 101
Xrt3dCreateWindow() 101
Xrt3dData structure 47, 103, 112, 113, 127
  application responsibility 49
  changing data 49
  convenience procedures 51
  description of 49
  example of setting 51
  input from file 48
  irregular grid 50
  overview 47
  regular grid 49
  use of as 4D surface 65
Xrt3dDataCopy() 101
Xrt3dDataShaded() 102
Xrt3dDataSmooth() 102
Xrt3dDataType 127
Xrt3dDataWindow() 102
Xrt3dDestroyData() 103
Xrt3dDetachWindow() 103
Xrt3dDistnIndex() 103
Xrt3dDistnMethod 127
Xrt3dDistnTable structure 70, 81, 103, 104, 105, 128
Xrt3dDistnTableFromFile() 103
Xrt3dDistnTableToFile() 104
Xrt3dDrawFormat 128
Xrt3dDrawScale 128
Xrt3dDrawToClipboard() 104
Xrt3dDrawToDC() 45, 104
Xrt3dDrawToFile() 45, 105
Xrt3dDupContourStyles() 14, 74, 105
Xrt3dDupDistnTable() 14, 105
Xrt3dDupStrings() 14, 106
Xrt3dDupValueLabels() 14, 106
Xrt3dDupXYColors() 14, 106
Xrt3dFreeContourStyles() 106
Xrt3dFreeDistnTable() 106
Xrt3dFreePropString() 13, 106
Xrt3dFreeStrings() 107
Xrt3dFreeValueLabels() 107
Xrt3dFreeXYColors() 107
Xrt3dGetAction() 59, 107
Xrt3dGetActionList() 59, 108
Xrt3dGetDistnTable() 108
Xrt3dGetHandle() 109
Xrt3dGetNthContourStyle() 75, 109
Xrt3dGetNthDataLabel() 109
Xrt3dGetNthFooterString() 109
Xrt3dGetNthHeaderString() 110
Xrt3dGetNthLegendString() 110
Xrt3dGetPalette() 110
Xrt3dGetPropString() 12, 13, 110
Xrt3dGetValueLabel() 39, 111
Xrt3dGetValues() 12, 111
Xrt3dGetWindow() 111
Xrt3dGetXYColor() 31, 111
Xrt3dGridData structure 49, 128
Xrt3dInterpMethod 128
Xrt3dIrGridData structure 50, 128
Xrt3dLegendStyle 129
Xrt3dLinePattern 73, 129
Xrt3dMakeDataFromFile() 50, 112
Xrt3dMakeGridData() 50, 113
Xrt3dMakeIrGridData() 113
Xrt3dMap() 62, 113
Xrt3dMapResult structure 62, 129
Xrt3dModifyCallbackStruct structure 129
XRT3DN_MODIFY_END message 121
XRT3DN_MODIFY_START message 55, 56, 60, 121
XRT3DN_PALETTECHANGED message 44, 121
XRT3DN_PROPERTIES message 122
XRT3DN_REPAINTED message 122
XRT3DN_RESIZED message 122
XRT3DN_ROTATE message 123

XRT3DN_TRANSFORM message 57, 123
Xrt3dPick() 61, 114
Xrt3dPickResult structure 61, 129
Xrt3dPreviewMethod 129
Xrt3dPrint() 45, 114
Xrt3dPropertiesCallbackStruct structure 129
Xrt3dRegion 130
Xrt3dResetContourStyles() 114
Xrt3dResizeCallbackStruct structure 130
Xrt3dRotateCallbackStruct structure 58, 130
Xrt3dSaveDataToFile() 115
Xrt3dSetAction() 60, 115
Xrt3dSetNthContourStyle() 115
Xrt3dSetNthDataLabel() 116
Xrt3dSetNthFooterString() 116
Xrt3dSetNthHeaderString() 116
Xrt3dSetNthLegendString() 116
Xrt3dSetPropString() 12, 13, 40, 43, 117
Xrt3dSetValueLabel() 39, 117
Xrt3dSetValues() 12, 43, 117
Xrt3dSetXYColor 31
Xrt3dSetXYColor() 117
Xrt3dStrokeFont 130
Xrt3dTextAttachMethod 133
Xrt3dTextCreate() 118
Xrt3dTextDestroy() 118
Xrt3dTextGetValues() 118
Xrt3dTextSetValues() 118
Xrt3dTransformCallbackStruct structure 57, 133
Xrt3dType 133
Xrt3dUnmap() 63, 119
Xrt3dUnpick() 62, 119
Xrt3dValueLabel structure 39, 133
Xrt3dXYColor structure 31, 133
Xrt3dZInterpolate() 119
Xrt3dZoneMethod 133

Z

zone charts 26
zooming, interactive 53