Motif対応XRT/3d™

プログラマーズ・ガイド&リファレンス・マニュアル

UNIX Edition ■ Version 3.0



株式会社タンジェント・コンピューティング
〒150-0001 東京都渋谷区神宮前6-6-11
tel:(03)3409-5364
fax: (03)3409-0147
info@tan.co.jp
http://www.tan.co.jp

1999年5月

本書は著作権法により保護されています。(株)タンジェント・コンピューティングによる事前の許可なく、本書のいかなる部分も無断転載、複製、複写を禁じます。
Copyright©1999 (株)タンジェント・コンピューティング

XRT、XRT/3d、XRT/care、XRT/field、XRT/gear、XRT/graph、XRT/table、XRT PDSはKL Group Inc.の登録商標です。
その他製品名、名称、サービス等は該当する会社または組織の商標または登録商標です。

XRT製品ライセンスについて

この度はXRT製品をご購入いただきまして誠にありがとうございます。

本製品のご利用に際しましては、次ページに記載の「XRTバイナリ使用許諾契約書」をご一読くださるようお願いいたします。下記に掲げる事項は、この契約の理解を補助するもので、契約の一部ではありません。

- XRT製品(XRT/graph、XRT/3d、XRT/table等)のライセンスは、機種(SUN SPARC、IBM RS/6000、HP 9000等)につき、またツールキット(Motif)につき許諾されます。
- XRT製品のライセンスにはシングルとネットワークの二種類があります。
- シングル・ライセンスでは、一つのCPUIDで識別されるマシンにおいてエンドユーザ・アプリケーションの開発にXRT製品をご使用いただくことができます。XRT製品を使用する開発には下記の内容が含まれます。
  - XRT製品のライブラリをエンドユーザ・アプリケーションのバイナリにコンパイル、リンクする。
  - XRT製品のビルダーを実行する。
  - XRT製品をGUIビルダーまたはUIMSツールに統合する。XRT製品が統合されているツールはXRT製品が使用許諾されているマシンでのみ使用できます。
- ネットワーク・ライセンスでは、ネットワークに接続されている同ビル内の同機種のマシンにおいてエンドユーザ・アプリケーションの開発にXRT製品をご使用いただくことができます。
- 静的にリンクされているXRT製品のライブラリを含むエンドユーザ・アプリケーションのバイナリは、ロイヤリティーやランタイム費用なしで提供していただくことができます。

XRTバイナリ使用許諾契約書

重要: ソフトウェア・パッケージを開封する前、またはソフトウェアをインストール、使用する前に必ずお読みください。

XRT製品の開封、インストール、使用、またはいずれかの左記の行為により、以下の規定につきお客様が承認されたものとさせていただきます。本契約書はお客様(個人または法人)とKL Group Inc.(以下、「KL Group」という。)の間で締結される法的な契約書です。もし、以下の規定に御同意頂けない場合には、ダウンロードやインストールされたファイルを直ちに破棄し、開封前のパッケージをライセンス費用の御支払いを示す書類と一緒に御返却ください。お客様がお支払になった料金は返却されます。

ソフトウェア・ライセンス

ライセンスの許諾 この使用許諾契約(以下、「ライセンス」という。)は、御指定のCPUと一致する一台のコンピュータにおいて、お客様が御購入されたXRTソフトウェア(以下、「ソフトウェア」という。)を御使用いただく権利をお客様に許諾するものです。ネットワーク・ライセンスを御購入された場合は、御指定のCPUと一致するコンピュータとローカル・エリア・ネットワークで接続された、同機種及び同ビル内に設置されているコンピュータにおいて、御購入いただいたライセンスで許可される使用台数を上限として「ソフトウェア」を御使用いただくことができます。また、共有ライブラリを御購入された場合、「ソフトウェア」のユーザ数は、御購入されたライセンスで許可された数を超えることはできません。「ソフトウェア」は、テンポラリ・メモリ(RAM等)にロードされた時、パーマネント・メモリ(ハード・ディスク、CD-ROM、その他の記憶装置等)にインストールされた時、エンドユーザ・アプリケーションにリンクされた時、開発ツール(コンパイラ、リンカー、GUIビルダー等)によってアクセスされた時に「使用」されたとします。他のコンピュータへ「ソフトウェア」のファイルを配布する目的のみで、ネットワークのファイル・サーバにインストールされている場合は、「使用」されているとはみなされません。

アップデート 本許諾契約書は、KL Groupから提供される「ソフトウェア」の全てのアップグレード版に適用されます。

著作権 「ソフトウェア」とそれに関連するイメージ、写真、テキスト等は、KL Groupとその供給者が所有するもので、その著作権は、カナダと米国の著作権法及び国際条約の規定で保護されています。従って、「ソフトウェア」は、書籍や音楽録音等、著作権で保護される他の製品と同じように扱ってください。「ソフトウェア」は、バックアップ目的で複製する以外は、一切複製作業を行うことはできません。また、「ソフトウェア」に付属する関連資料も同様に扱ってください。

その他の制限 「ソフトウェア」は、貸与、リース、譲渡、リバース・エンジニア、逆コンパイル、逆アセンブル、パスワードの作成をすることはできません。但し、これらの制限が司法管轄区域によって許可されていない場合は、この限りではありません。

「ソフトウェア」で構築されたアプリケーションの配布 「ソフトウェア」の"lib"または"lib-shared"ディレクトリにある、オブジェクト・コードのアプリケーション・ファイルにリンク可能な形態のファイル(以下、「XRTライブラリ・ファイル」という。)は、次の条件に従う場合にのみ、ロイヤリティなしで複製、配布することができます。(1)再リンクが不可能な方法でソフトウェアにリンクされて「XRTライブラリ・ファイル」が配布される。(2)配布されるソフトウェアは、「ソフトウェア」と同等または類似した機能を持つ開発ツールではない。(3)配布するソフトウェアの販売活動に、許可なくKL Groupの社名、ロゴ、トレード・マーク等を使用しない。(4)配布するソフトウェアには、著作権に関する注意書きを明記する。(5)配布したお客様のソフトウェアに対して第三者からなされた損害賠償請求に基づく損害等については、KL Group及びその供給者は一切その責任を負わない。それ以外の場合は、「ソフトウェア」のヘッダー・ファイルや(APIのドキュメンテーションを含む)ドキュメンテーション類を配布することはいかなる方法であってもできません。

U.S. Government Restricted Rights 「ソフトウェア」とその関連資料は、RESTRICTED RIGHTSに基づいて提供されています。本契約書で許諾される「ソフトウェア」とその関連資料は、Federal Acquisition Regulations (FAR) Part 12と Defense FAR Supplement (DFARS) 252.227-7014 (JUN 1995)で定義される"Commercial computer software"です。FAR 12.212 とDFARS 252.227.7014に基づき、この「ソフトウェア」の使用、複写、開示は、本契約書の条項に従います。製造者: KL Group Inc., 260 King Street East, Toronto, Ontario, Canada, M5A 1K3.

保証規定

品質保証 KL Groupは、「ソフトウェア」購入日より60日間、「ソフトウェア」がその関連資料記載の仕様に従って実質的に動作することを保証します。但し、品質保証の期間の制限が司法管轄区域によって許可されていない場合は、適用法例の範囲内でのみ有効です。尚、本品質保証は、第三者に譲渡することはできません。KL Groupによる故意、重大な過失により生じた損害、または製品に対する責任は、KL Groupは除外いたしません。

補償 KL Groupがその判断に基づいて行う補償は、次の事項に限定されます。(1)御支払いになられた金額を払い戻す。(2)上記記載の「品質保証」を満たさない「ソフトウェア」に対して補修、交換する。但し、(1)、(2)いずれの対応も、「ソフトウェア」と共ににその購入を証する書類を返却された場合にのみ提供されます。また、かかる事態が事故、故意若しくは過失、誤用等、KL Groupの責に帰さない理由により生じた場合は、本品質保証は無効になります。尚、交換された「ソフトウェア」の保証期間は、その受領より30日か、または交換の対象になった「ソフトウェア」の保証期間の残りか、どちらか長い方が適用されます。

権利侵害に対する補償 「ソフトウェア」が、第三者の特許、業務上の秘密、著作権等の侵害の対象となった場合、またはKL Groupがそのように判断した場合は、KL Groupは、次の補償を行うことがあります。(1)「ソフトウェア」を継続して使用できる権利を取得する。(2)「ソフトウェア」を権利侵害の対象にならないように交換、修復する。但し、交換または修復された場合も、以前と同じ機能は保つようにする。上記(1)及び(2)が適正でない場合は、(3)「ソフトウェア」のライセンス許諾を中止し、削除する。その場合、ライセンスに対して支払われた金額の内、定額法を適用した5年間の償却期間に基づいた相当金額を返却する。

保証の制限 司法管轄区域の適用法令の範囲内で、KL Group及びその供給者は、「ソフトウェア」とその関連資料の商品性、特定目的適合性の保証を含め、本保証規定に規定されていないその他の保証を、明示たると黙示たるとを問わず一切いたしません。この有限保証は、お客様に特定な法的権利を与えるものですが、州、管轄区域によっては、その他の権利が与えられることもあります。。

免責 司法管轄区域の適用法令の範囲内で、KL Group及びその供給者は、「ソフトウェア」の使用または使用不能から生ずる損害(事業利益の損失、事業の中断、事業情報の損失または金銭的損害を含むがこれらに限定されない)に関して、一切責任を負わないものとします。例え、KL Groupがかかる損害の可能性について知らされていた場合も同様です。但し、州や管轄区域によっては、必然的または偶発的な損害に対する責任の除外や制限を認めていない場合があり、その場合、前述の制限は適応されないことがあります。

適用法令 お客様がEuropean Unionに属する地域に居住されている場合、本契約書はその国の法律に準拠し、契約当事者双方は、その国の司法に従います。それ以外の場合は、本契約書は、カナダ・オンタリオ州の法律に準拠し、契約当事者双方は、オンタリオ集の司法に従います。

パブリック・ドメイン・ソフトウェア 本契約書に記載されている条項は、パブリック・ドメインと明記されているソフトウェアには適用されません。

本契約書は、KL Groupより提供される"XRT FOR MOTIF BINARY LICENSE AGREEMENT"を(株)タンジェント・コンピューティングが日本語訳したものです。表現等に不一致がある場合は、"XRT FOR MOTIF BINARY LICENSE AGREEMENT"(次ページ参照)に従います。

XRT FOR MOTIF BINARY LICENSE AGREEMENT

Important — Read Carefully Before Opening Software Package(s), Installing or Using Software On Your Computer

By opening a sealed software package, or installing, or using this XRT product, you indicate your acceptance of the following XRT for Motif Binary License Agreement. This is a legal agreement between you (either an individual or an entity) and KL Group Inc. If you do not accept and agree with all of the terms of this agreement, promptly destroy any downloaded or installed files, and return the unopened software package(s) and all other materials with proof of payment to your place of purchase, and your license fee will be refunded.

SOFTWARE LICENSE

Please Note: This is a license agreement and not an agreement for sale.

Grant of License. This License Agreement ("License") permits you to use one copy of the software you purchased (the "SOFTWARE"), on a single computer after registration of that computer's CPU identification with us. If you have purchased a network site license you may also use the SOFTWARE on computers in the same building having the same architecture as a registered computer, and connected by the same local area network, provided that the total number of such computers, on which the SOFTWARE is ever used, does not exceed the maximum number permitted by the network site license which you ordered.

If you have purchased shared library versions of the SOFTWARE, the number of users of the SOFTWARE cannot exceed the maximum number permitted by the license which you ordered.

The SOFTWARE is "in use" on a computer when it is loaded into temporary memory (i.e. RAM); when it is installed into permanent memory (e.g. hard disk, CD-ROM, or other storage device); when it is being linked into an end-user application; or when it is accessed from a software development tool such as a compiler, a linker, or a GUI builder. The SOFTWARE is not considered "in use" when installed on a network file server for the sole purpose of file distribution to other computers.

Updates. This agreement will govern the terms of any SOFTWARE updates provided to you by KL Group.

Copyright. The SOFTWARE (including any images, photographs and text incorporated into the SOFTWARE) is owned by KL Group or its suppliers and is protected by Canadian and United States copyright laws and international treaty provisions. Therefore you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) and not reproduce it except that you may make a backup for archival purposes. You may not copy the printed materials accompanying the SOFTWARE.

Other Restrictions. You may not rent, lease, transfer, reverse engineer, decompile, disassemble, create passwords for or translate the Software, except to the extent such foregoing restriction is expressly prohibited by applicable law.

Distribution of applications you build with the SOFTWARE. You have a royalty-free right to reproduce and distribute files in the SOFTWARE that are found in the "lib" or "lib-shared" directories and which are in a form suitable for linking into object code application files (the "XRT Library Files") provided that (a) you distribute the XRT Library Files only when linked into your software product in such a way as to prohibit further relinking; (b) your software product is not an application development tool that provides similar functionality to the SOFTWARE licensed hereunder; (c) you do not use KL Group's name, logo or trademark to market your software product; (d) you include a valid copyright notice on your software product; and (e) you agree to indemnify, hold harmless, and defend KL Group and its suppliers from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software product. Unless otherwise specified, you may not in any manner distribute any header files or documentation (including API documentation) provided as part of the SOFTWARE.

U.S. Government Restricted Rights. The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. The SOFTWARE (including documentation) licensed in this agreement is "Commercial computer software" as defined in Federal Acquisition Regulations (FAR) Part 12 and Defense FAR Supplement (DFARS) 252.227-7014 (JUN 1995). In accordance with FAR 12.212 and DFARS 252.227.7014, notwithstanding any provision to the contrary, use, duplication and disclosure of this software is governed by the terms of this agreement. Manufacturer is KL Group Inc., 260 King Street East, Toronto, Ontario, Canada, M5A 1K3.

LIMITED WARRANTY

Limited Warranty. KL Group warrants that the SOFTWARE will perform substantially in accordance with the accompanying written materials for a period of sixty (60) days from the date of receipt. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. This warranty may not be assigned. KL Group does not exclude product liability or liability for damages resulting out of intent or gross negligence on the part of KL Group or its leading personnel.

Customer Remedies. KL Group's and its suppliers' entire liability and your exclusive remedy shall be, at KL Group's option, either (a) return of the price paid, or (b) repair or replacement of SOFTWARE that does not meet KL Group's Limited Warranty and which is returned to KL Group with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

Customer Remedies for Infringement. Should the SOFTWARE become or in KL Group's opinion be likely to become the subject of a claim for infringement of patent, trade secret or copyright, KL Group may (a) procure for you the right to continue to use the SOFTWARE; or (b) replace or modify the SOFTWARE to make it non-infringing, provided that the same function is performed by the replacement or modified SOFTWARE; or if (a) or (b) are not commercially reasonable, (c) terminate the license to use the SOFTWARE, remove the SOFTWARE, and grant you a credit for the license fee as depreciated on a straight-line five (5) year basis.

No Other Warranties or Obligations.. To the maximum extent permitted by applicable law, KL Group and its suppliers shall have no other obligations and disclaim all other warranties, either express or implied, including, but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the SOFTWARE and the accompanying printed materials. This limited warranty gives you specific legal rights. You may have others which vary from state/jurisdiction to state/jurisdiction.

No Liability for Consequential Damages. To the maximum extent permitted by applicable laws, in no event shall KL Group or its suppliers be liable for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of information, or other pecuniary loss) arising out of the use of or inability to use this KL Group product, even if KL Group has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

Governing Law. If you are a resident of one of the member states of the European Union, this Agreement is governed by the laws of your country of residence, and each of the parties hereto irrevocably attorns to the jurisdiction of the courts of your country of residence. Otherwise, this Agreement is governed by the laws of the Province of Ontario, Canada. Each of the parties hereto irrevocably attorns to the jurisdiction of the courts of the Province of Ontario.

Public Domain Software. This license agreement and the warranties, remedies, limitations, conditions and restrictions contained herein do not apply to any software clearly marked as being in the public domain.

BINLIC-M-11/98

目次

はじめに 1

XRT/3dについて 1
XRT/3d 3.0での変更点 2
XRT/3d 3.0の新しい機能 2
前提知識 3
表記上の規約 3
関連資料 3
テクニカル・サポート 4
お客様のご意見 4

Part I　XRT/3dの使用法

1　XRT/3dの 簡単なプログラムの開発 9

1.1　はじめに 9
1.2　環境変数 9
1.3　コンパイルとプログラムの許可 10
1.4　simple.c プログラム 11
1.5　simple.cの3Dデータ 12
1.6　simple.cのコントロール 13
1.7　サンプル・コード 13

2　XRT/3dの基本操作 15

2.1　Motifとの統合 15
2.2　用語の解説 15
2.3　浮動小数点リソース 17
2.4　UseDefaultリソース 18
2.5　ポインタ・リソース 18
2.6　リソース・アップデートのバッチ処理 19
2.7　色 19
2.8　リソース・ファイル 21
2.9　デバッグ 21

3　XRT/3dのプログラミング 23

3.1　リソース 23
3.2　軸のコントロール 29

3.3 凡例の表示 29
3.4 遠近法 31
3.5 軸のスケーリング 31
3.6 軸の注釈 32
3.7 グリッド線 33
3.8 ヘッダーとフッターのテキスト 33
3.9 ヘッダー、フッターと凡例のフォント 34
3.10 領域の位置決め 35
3.11 領域の枠線 36
3.12 前景と背景の色 36
3.13 背景のイメージ 38
3.14 ダブル・バッファリング 40
3.15 出力と印刷 40

4 XRT/3dのプログラミング: サーフェスと棒 43

4.1 サーフェスと棒グラフの15の基本タイプ 43
4.2 棒グラフとヒストグラム 48
4.3 等高線とゾーンの表示 52
4.4 メッシュのコントロール 54
4.5 サーフェスの色 55
4.6 サーフェスの塗りつぶし 55

5 XRT/3dのプログラミング: スキャッター・プロット 57

5.1 スキャッター・プロットの三つの基本タイプ 57
5.2 シンボルとドロップ・ラインのスタイル 58
5.3 データ・スタイル 58

6 XRT/3dデータ 63

6.1 データについて 63
6.2 Xrt3dData構造体 66
6.2.1 規則的なグリッドのデータ 66
6.2.2 不規則的なグリッドのデータ 67
6.2.3 スキャッター・プロットのデータ 68
6.2.4 簡易手続き 69

7 ユーザ・インタラクションのプログラミング 71

7.1 デフォルトのユーザ・インタラクション 71
7.2 トランスレーションとアクションの概要 73
7.3 ユーザ・インタラクションの開始 73
7.4 ユーザ・インタラクションのアップデート 74
7.4.1 スケーリング、移動、ズーム 74

7.4.2 回転 76
7.4.3 インタラクションの不許可 77
7.5 ユーザ・インタラクションの終了 78
7.6 グラフ・データとのインタラクション 78
7.6.1 アクション・ルーチンとコールバック 78
7.6.2 イベント・ハンドリングの手続き 80
7.7 ウィンドウのリサイズ 84

8 高度なプログラミング 87

8.1 4Dサーフェス・グラフ 87
8.2 4D棒グラフ 88
8.3 テキスト・オブジェクト 89
8.4 分布表(Distribution Table)のカスタマイズ 93
8.5 凡例ラベルのカスタマイズ 94
8.6 等高線のカスタマイズ 95
8.7 国際化サポート 98
8.8 MS Windowsへのポーティング 98
8.9 C++の使用 101
8.10 UILの使用 101
8.11 リソース値をストリングに変換する 102

9 PDS ユーティリテイ・ライブラリ 105

9.1 XrtPDSImage 105
9.1.1 XrtPDSImageの作成 105
9.2 Image 型の登録 106
9.2.1 定義済みのイメージ 106
9.2.2 任意のイメージ 107
9.2.3 イメージ・ファンクションの書き込み 109
9.3 ダイナミックまたは共用ライブラリの使用 110
9.4 XrtPDSListオブジェクト 110
9.4.1 サンプル・コード 110
9.4.2 リストの作成と破棄 111
9.4.3 リストのアイテムの追加と削除 111
9.4.4 アイテムの検索と取得 113
9.4.5 リスト比較ルーチン 115
9.4.6 リストのソート 116
9.4.7 ユーザが定義したデータ 117
9.5 手続きと方法 118
9.5.1 XrtPDSImageの方法 118
9.5.2 XrtPDSListオブジェクトの方法 120

Part II リファレンス

A リソースの一覧 133

A.1 ウィジェットの概要 133
A.2 XRT/3dのリソース 134
A.3 継承されるリソース 156

B テキスト・オブジェクトの リソースの一覧 157

B.1 オブジェクトの概要 157
B.2 テキスト・オブジェクトのリソース 157
B.3 継承されたリソース 160

C 手続きと方法 161

D トランスレーションとアクション 187

D.1 デフォルトのトランスレーション 187
D.2 アクション・ルーチン 188

E データ型 193

F リソース名、型とクラス 205

G リソース・ファイルの使用 217

G.1 構文 217
G.2 リソース・ファイルの例 220

H UIL引数 223

索引 229

はじめに

XRT/3dについて ■ XRT/3d 3.0での変更点

XRT/3d 3.0の新しい機能 ■ 前提知識

表記上の規約 ■ 関連資料

テクニカル・サポート ■ お客様のご意見

XRT/3dについて

XRT/3dウィジェット

XRT/3dは、ボタン、スライダ、メニュー等のウィジェットから構成されているMotifに新たなウィジェットを加えます。このグラフ・ウィジェットは3Dデータをサーフェス、等高線、棒グラフ等に表示します。

XRT/3dウィジェットには、グラフの表示や動きをコントロールするリソースが提供されています。ウィジェットが一つ増えた点を除けば、XRT/3dを使用したプログラミングは、他のMotifのプログラミングとほとんど違いはありません。

XRT/3dは、XRTファミリのXRT/graphやXRT/tableと組み合わせても使用することができます。XRT/graphは2Dデータを線、棒、円グラフ等に表示し、XRT/tableは、行と列の表形式のデータを表示します。

XRT/3dには、下記の内容をコントロールするリソースがあります。

- 描画(メッシュ、サーフェス、等高線、ゾーン)とグラフ・タイプ(サーフェス、棒グラフ)
- 等高線のスタイル(線幅、色、パターン、塗りつぶし)
- 分布(distribution)の方法、分布表、分布レベルの数
- 隠線の表示・非表示、サーフェス/メッシュの色
- 3D回転と遠近法
- 凡例の位置、方向、スタイル、枠線スタイル、アンカー、フォント、色
- ヘッダー/フッターの位置、枠線スタイル、テキスト、フォント、色
- グラフの位置、枠線スタイル、色、幅、高さ
- 軸の最小、最大、フォント、グリッド線、タイトル、ラベル
- ウィンドウの背景/前景色

XRT/3dには、次の作業を行う手続きや方法が提供されています。

- グラフ化される数値を含んだデータ構造体の割り当てやロード
- グラフのPostScriptやCGMフォーマットへの出力
- ユーザ・イベントの対応支援
- エンドユーザによる3Dビューのインタラクティブ回転、ズーム、シフト
- インデックス付きリソースの設定、取得支援

XRT/3d 3.0での変更点

XRT/3d 3.0は、XRT/3d 2.1と互換性があります。既存のアプリケーションでXRT/3d 3.0を使用する場合は、次の点に留意してください。

リコンパイル	**このリリースのXRT/3dを使用するには、既存のアプリケーションを(再リンクだけではなく)再コンパイルしてください。**

XRT/3d 3.0の新しい機能

スキャッター・プロット

三つのデータ・ポイント(x, y, z)からスキャッター・プロットを作成する新しいグラフのタイプとデータ型が追加されました。(P57参照)

グリッド線

新しいデータ型、`Xrt3dLineStyle`により、色、幅、線スタイルなどを定義できるようになりました。デフォルトのスタイルは、グラフの前景色(指定されていなければウィジェットの前景色)を使用した幅1の実線です。(P33参照)

凡例のコントロール

新しいリソースにより、凡例のレベルに対してインターバルを指定できるようになりました。また、凡例のタイトルのリソースも追加されました。(P29参照)

背景のイメージ

ウィジェット、グラフ、ヘッダー、フッター、凡例、テキスト・オブジェクトの背景にはイメージを表示することができます。(P38参照)

PDSユーティリティ・ライブラリ

PDSユーティリティ・ライブラリを利用して、グラフをJPEG、PNGまたは、ユーザが定義したイメージ・フォーマットに出力できるようになりました。XrtPDSImageは、イメージを持つオブジェクトです。Pixmapが内部で使用されますが、この新しい方法によって、様々なイメージのフォーマットをサポートすることができます。

このライブラリは、その他、XrtPDSListオブジェクトもサポートしています。これを利用して、複数のアイテムのグループを簡単に記憶させることができます。

前提知識

このマニュアルは、ユーザがC言語及びMotifツールキットに精通していることを前提としています。MotifやXRT/3dアプリケーションをプログラミングするには、C言語の“char *配列”、“構造体のポインタ”等を理解している必要があります。またリソース値の取得や設定等のMotifの基本的な概念も理解していなければなりません。Motifに関する資料については“関連資料”をご覧ください。

表記上の規約

`Typewriter Font`

- Cコードやファイル内容の例
- XRT/3dやMotifリソース名、定数
- 端末で入力するコマンド

斜体

- パス、ファイル名、プログラム、手続き、パラメータ
- 新しい用語や重要な用語
- 図やテーブル
- このマニュアルで参照されている参考資料名(Motif Programming Manual等)

`...[XYZ]...`

の情報を指定するXRT/3dのリソースは、X、Y、Z軸、どれも同様です。リソース名の[XYZ]は、X軸、Y軸、Z軸に対応するリソースを意味します。例えば"`XtNxrt3dXMin`"、"`XtNxrt3dYMin`"、"`XtNxrt3dZMin`"は"`XtNxrt3d[XYZ]Min`"となります。

関連資料

*Motif Programming Manual*と*Motif Reference Manual*(O'Reilly & Associates,Inc出版)

X Toolkit Intrinsics Reference Manual、*Xlib Programming Manual*、*Xlib Reference Manual*(O'Reilly & Associates,Inc.出版) これらのマニュアルはXRT/3dやMotifの開発には直接必要ありませんが、Xlibの知識はX Windowアプリケーションの開発に重要です。

テクニカル･サポート

初期の問題の多くは、インストールやセットアップに関する問題です。リリースノート&インストールガイドには、インストールやセットアップの際の問題の回避法がありますので参照してください。

XRTテクニカル･サポートへお問い合わせされる場合は、次の事項をご連絡ください。

- XRT/3dライセンスのシリアル番号
- XとMotifのバージョン
- OSのバージョン

オンライン･サポート

(株)タンジェント･コンピューティング及び、開発元のKL Group Inc社のwebサイトには、技術的な情報が掲載されていますので参照してください。

技術情報及びFAQ **(日本語)**	http://www.tan.co.jp/xrt/support/
技術情報(英語)	http://www.klgroup.com/cs/tech/xrt/
Knowledge Base(**英語**)	http://www.klgroup.com/cgi-bin/xrt_search.cgi
FAQ(**英語**)	http://www.klgroup.com/cs/tech/xrt/faq/

尚、テクニカル･サポートを御受けいただくには、XRT/careと呼ばれる、保守サポート･サービスを御購入されていることが前提となります。XRT/careのお申し込み法やサービスに関する御質問等がございましたら、(株)タンジェント･コンピューティングまでご連絡ください。

お客様のご意見

XRT製品のアップグレードで追加される新しい機能の大半は、お客様のリクエストに基づいています。XRT/graphに関するお客様のご意見、ご使用例、或は、あれば便利と思われる機能等がございましたら下記までご連絡ください。

株式会社タンジェント･コンピューティング
〒150-0001東京都渋谷区神宮前6-6-11
電話 03-3409-5364
Fax 03-3409-0147
Email info@tan.co.jp
WWW http://www.tan.co.jp/

Part I

Using XRT/3d

1

XRT/3dの簡単なプログラムの開発

はじめに ■ 環境変数

コンパイルとプログラムの許可 ■ simple.c プログラム

simple.cの3Dデータ ■ simple.cのコントロール

サンプル・コード

1.1 はじめに

この章では、*simple.c*というXRT/3dのプログラミングを実際にコンパイル、実行しながら、XRT/3dの基本的なプログラミングを紹介します。ユーザは、XRT/3dの最も大切なグラフのコントロール、DrawMesh、DrawShaded、DrawContours、DrawZonesを操作することができます。

このプログラムは*$XRTHOME/src/3d/examples/simple.c*にあります。

1.2 環境変数

最初にUNIXの環境をご確認ください。*XRTHOME*という環境変数が、XRT/3dのファイルのディレクトリ名に定義されていなければなりません。

次のように入力して、XRTHOMEの現在の値を確認できます。

```
echo $XRTHOME
```

もしXRTHOME環境変数が正しく設定されていない場合は、リリース&インストレーション・ガイドを参照するか、システム管理者の方にお問い合わせください。

またMotifのアプリケーションがコンパイルできることもご確認ください。ご使用になっているMotifのマニュアルで、Motifのライブラリとインクルード・ファイルの位置を調べることができます。XRT/3dを使用される前に、簡単なMotifのアプリケーションがコンパイルでき稼動することも必ずご確認ください。

1.3 コンパイルとプログラムの許可

X、Motif、XRT/3dのライブラリとインクルード・ファイルが標準の場所(*/usr/lib*、*/usr/include/X11*、*/usr/include/Xm*)にインストールされていれば、*simple.c*は次のようにして、コンパイル[1] や許可を与えることができます。

```
cc   simple.c -I$(XRTHOME)/include \
  -L$(XRTHOME)/lib -o simple \
  -lxrt3d -lpdsutil -lXm -lXpm -lXt -lX11 -lm

$XRTHOME/bin/xrt_auth simple
```

もし、ライブラリとインクルード・ファイルが標準の位置になければ、`-I`と`-L`のコンパイラ・フラグが必要です。

xrt_auth

XRT/3dのライブラリが実行可能アプリケーションにリンクされたら、*xrt_auth*を使用して許可を与えます。*xrt_auth*はご使用のマシンにXRT/3dのライセンスがあるかを調べ、アプリケーションに許可を与えます。*xrt_auth*には、`-v`(verbose message;詳細メッセージ)と`-q`(quiet mode;メッセージ出力なし)の二つのフラグのオプションがあります。

```
xrt_auth [-q | -v] executable_file
```

エンドユーザ・アプリケーションは、*xrt_auth*により許可が与えられると、どのマシンでも実行することができます。XRT/3dで作成されたアプリケーションにはランタイム費用はかかりません。

アプリケーションが*xrt_auth*によって許可が与えられないと、グラフは数秒後に消滅し、図1のメッセージが現われます。また、トラブルシューティングに関するメッセージが*stdout*に書かれます。



図1　期限切れまたは未許可の場合のメッセージ

[1] コンパイルのオプションは機種に依存します。ご使用のマシンでのコンパイルの例が*$XRTHOME/src/3d/examples*の*Makefile*にあります。

注意: 共用ライブラリのリンクや許可に関しては、リリースノートを参照してください。

C++とANSI C

XRT/3dのインクルード・ファイルはC++と、ANSI C、"Kernighan and Ritchie" Cと互換性があります。

1.4 simple.c プログラム

この章では、*simple.c*というXRT/3dのプログラムを使用します。このプログラムとMakefileは、*$XRTHOME/src/3d/examples*にあります。

*simple.c*は約150行のプログラムです。そのうち約20%は、formウィジェットの上部にあるトグル・ウィジェット等の一般的なMotifのウィジェットに関するコードです。ここでは、それ以外のXRT/3dに関係するコードについて解説します。

*simple.c*がコンパイル、実行されると、図2のウィンドウが表示されます。このセクションの先に進む前に、*simple.c*をコンパイル、実行させてみてください。

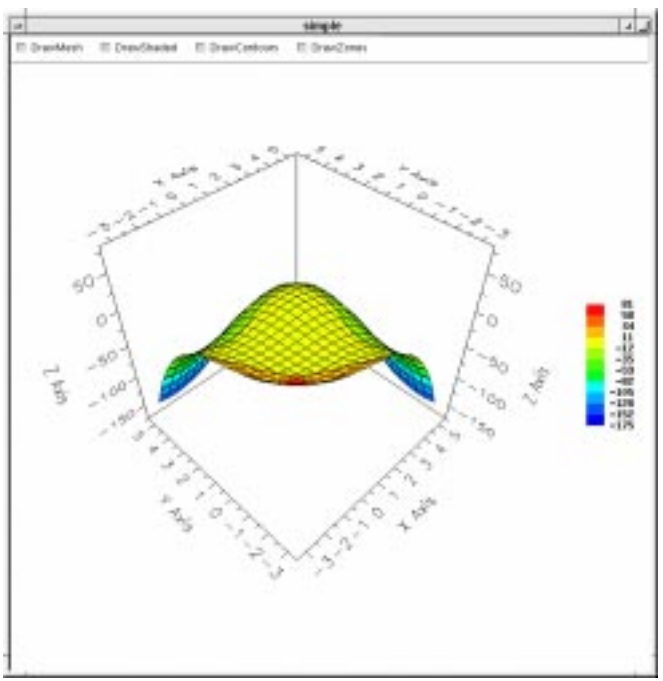


図2　*simple.c*によって作成されたウィンドウ

*simple.c*は、XRT/3dプログラミングの幾つかの基本を示しています。ほとんどの初期化は*CreateWidgets()*により行われています。ウィンドウが生成され、上部にはチェック・ボックス、その下にはXRT/3dウィジェットが作成されています。このウィンドウがリサイズされれば、グラフも自動的にリサイズします。

1.5 simple.cの3Dデータ

XRT/3dでグラフを表示するには、データをXrt3dData構造体にロードする必要があります。データは、ファイル、データベース、他のマシンで実行されている処理等、あらゆるソースのデータが使用できます。*simple.c*は、Xrt3dData構造体を作成し，次の式によって定義されるサーフェスのグリッド･データをロードします。

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

XRT/3dデータの詳しい説明は、6.0(P63)にあります。ここでは、先に定義したサーフェス上のX軸、Y軸に沿ったグリッドを解説します。Xのグリッド線がYのグリッド線と交差する時、その交差点がサーフェスの点を定義します。この点において、サーフェスの式の数値を求め、その解をXRT/3dウィジェットに渡し、グラフを作成します。

始めに、グリッドの点の数を決めます。点の数が少ないとサンプル数が不足し、解像度が低いイメージのグラフになります。また点が多すぎると、画面上には反映されず、余計なオーバーヘッドがかかりますので注意してください。

*simple.c*は、*CalculateGrid()*でXrt3dData構造体を定義します。numxとnumyの値は両方とも20に設定され、これらがグリッドの解像度を決定します。*simple.c*は、*Xrt3dMakeGridData()*を呼び出し、Xrt3dData 構造体に20 x 20 Z値に十分なスペースを割り当てます。

```
grid = Xrt3dMakeGridData(numx, numy,
 XRT3D_HUGE_VAL, 8.0/(numx-1), 8.0/(numy-1),
 -3.0, -3.0, True);
```

*Xrt3dMakeGridData()*が成功すれば、Xrt3dData構造体のポインタを返します。失敗時にはNULLを返します。最初のnumxとnumyのパラメータは、割り当てる構造体の大きさを示します。

三番目のパラメータは、“no value”を意味するデータ値を指します。このサーフェス･データにはhole(穴)がないので、これをXRT3D_HUGE_VALにセットします。

次の二つのパラメータは、X、Y方向のグリッド線のstep(間隔)を指定します。ここで作成するグリッド線は、-3から5までに20本の線があるので、間隔は8.0/(20-1)または約0.421になります。

その次の二つのパラメータは、データの原点(-3, -3)の位置です。最後のパラメータは、*Xrt3dMakeGridData()*にZ値のスペースを割り当てるかどうかを指示します。ここでは、XRT/3dにスペースを割り当てさせるように、Trueにセットします。

これで、Xrt3dData構造体がグリッド･データを持つことができるようになりました。次は、各グリッド･ポイントでサーフェスの式の数値を求めます。

Xrt3dDataの*g.values*エレメントは、*double*型の２次元配列です。次のコードは、サーフェスのグリッド値でこの構造体をセットします。

```
for (i = 0; i <numx; i++) {
  x = grid->g.xorig + i*grid->g.xstep;
  for (j = 0: j < numy; j++) {
 y = grid->g.yorig + j*grid->g.ystep;
 grid->g.values[i][j] = 3*x*y - x*x*x - y*y*y;
  }
}
```

グリッドで参照されるXrt3dData構造体は、これでグラフを作成する準備ができました。*simple.c*がXRT/3dウィジェットを作成すると、XtNxrt3dSurfaceDataのリソースがgridにセットされ、ウィジェットにデータが接続されます。

1.6 simple.cのコントロール

*simple.c*のウィンドウの上部には、ユーザが操作できるトグルがあります。

DrawMesh、DrawShaded、DrawContours、DrawZonesのトグルはそれぞれ、XtNxrt3dDrawMesh、XtNxrt3dDrawShaded、XtNxrt3dDrawContours、XtNxrt3dDrawZonesのリソースをオンまたはオフにします。

これらの四つのリソースから、基本的な3Dデータ表示の15種類の組み合わせが作成できます。これらのリソースは3.2に解説があります。

ほとんどのリソースは次に続く章で解説され、Appendix Aには全てのリソースの一覧がありますので参照してください。

1.7 サンプル･コード

XRT/3dには、幾つかの例やデモのプログラムが提供されています。例は、特定な機能を使用した短いプログラムです。デモは、複数の機能を使用した、アプリケーションです。

例

これらの例プログラムは、*$XRTHOME/src/3d/examples*にあります。

- *simple.c*
- *bars.c*

texts.c

scatter.c

- *simple_cpp.cxx*
- *simple_uil.c*

デモ・プログラム

デモ・プログラムは、*$XRTHOME/src/3d/demos*にあります。各デモごとにディレクトリがあります。

- ***feedback3d***は、マップとピックのフィードバック機能を使用します。ユーザはサーフェスのグリッド・ポイントをピックし、そのポイントを新しい値にドラッグできます。
- ***math3d***は、提供されている関数をグラフ化します。任意の関数を入力したり、グラフの表示を変更することができます。
- ***play3d***は、XRT/3dとテキスト・オブジェクトの全てのリソースに直接にアクセスします。このプログラムは、リソース・ファイルにグラフの記述をロード、保管することができ、このファイルはアプリケーションで使用することができます。
- ***shader***は、サーフェスにシェーディング(影)を加えることによりビジュアル効果を強化します。このプログラムは、*Xrt3dDataShaded()*を使用し、光源の強さや角度に従ってサーフェスの各面にシェーディングを加えます。
- ***spline***は、キュービックまたはリニア・スプライン補間を使用して、大きなデータセットのサイズを小さくする例です。サイズを小さく保つことにより、XRT/3dのパフォーマンスが向上します。
- ***view_dir***は、3Dデータをヒストグラムで表示します。このプログラムは選択されたディレクトリにあるファイルの数やサイズをビジュアル化します。

複数のXRT製品を使用したデモ

- ***mvc***は、XRT/graph、XRT/table、XRT/3dを使用した、XRT MVC Kitのデモです。
- ***profile***は、XRT/3dによって表示される3Dサーフェスの2D断面図をXRT/graphで表示します。
- ***surface_editor***は、XRT/3dとXRT/tableを一つのアプリケーションで使用している例です。

2

XRT/3dの基本操作

Motifとの統合 ■ 用語の解説

浮動小数点リソース ■ UseDefaultリソース

ポインタ・リソース ■ リソース・アップデートのバッチ処理

色 ■ リソース・ファイル

デバッグ

2.1 Motifとの統合

XRT/3dはMotifツールキットに新たなウィジェットを追加します。このウィジェットは、他のMotifプログラミングと同様、Xt Intrinsicsで操作することができます。

Xt Intrinsicsは、ウィジェットをクラス編成するオブジェクト指向アーキテクチャを定義します。例えば、XmPushButtonウィジェットは、Xt Coreウィジェットのサブクラスの XmPrimitiveのサブクラスのXmLabelのサブクラスであり、XRT/3dウィジェットは XmPrimitiveのサブクラスとなります。これは、XRT/3dウィジェットで定義されるリソースの他にXmPrimitiveとCoreのリソースも設定できることを意味します。例えば、(XmPrimitiveで定義される)`XmNborderWidth`リソースは、XRT/3dウィジェットを囲む枠線幅を設定することができます。また、(Coreで定義される)`XtNwidth`と`XtNheight`リソースは、XRT/3dウィジェットの幅と高さを設定することができます。

2.2 用語の解説

図4は、XRT/3dウィジェットの構成を示しています。ウィジェットには、ヘッダー、フッター、凡例、グラフの四つの領域があり、各領域は、それぞれ起点、幅と高さを持っています。

ユニット・キューブ

*ユニット・キューブ(unit cube)*は、(軸も含む)3D全体を囲む最小単位のキューブ (立方体)として定義されます。軸のスケーリングやフォントのサイズを指定するリソース等、XRT/3dの幾つかのリソースは、このユニット・キューブのサイズに依存します。

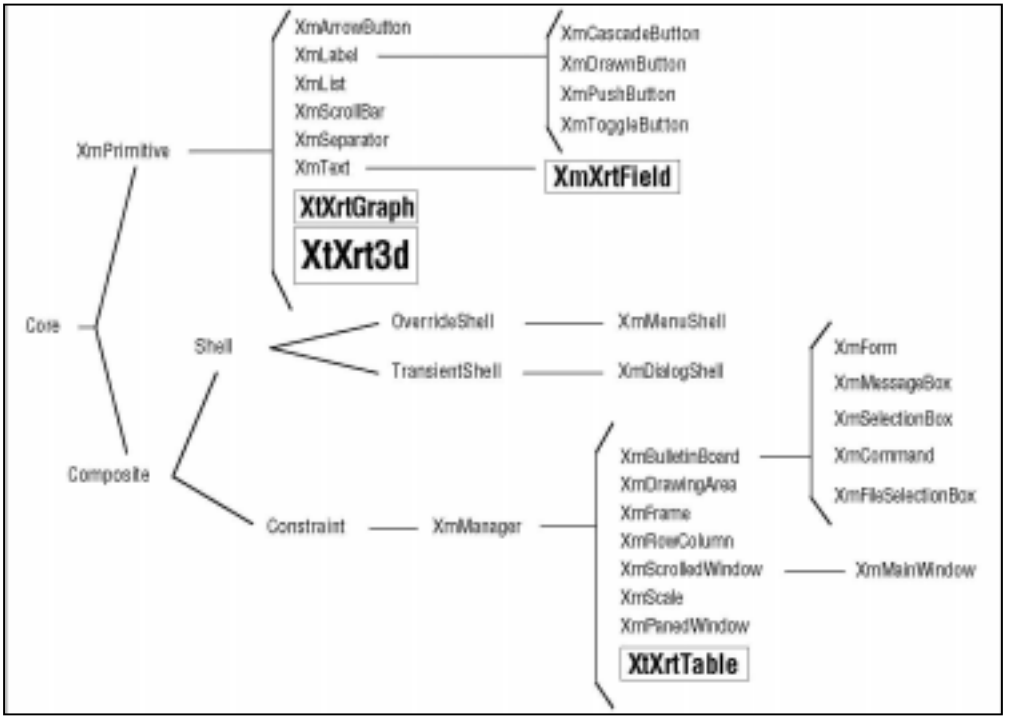


図3　Motifクラス階層内のグラフ・ウィジェットの位置

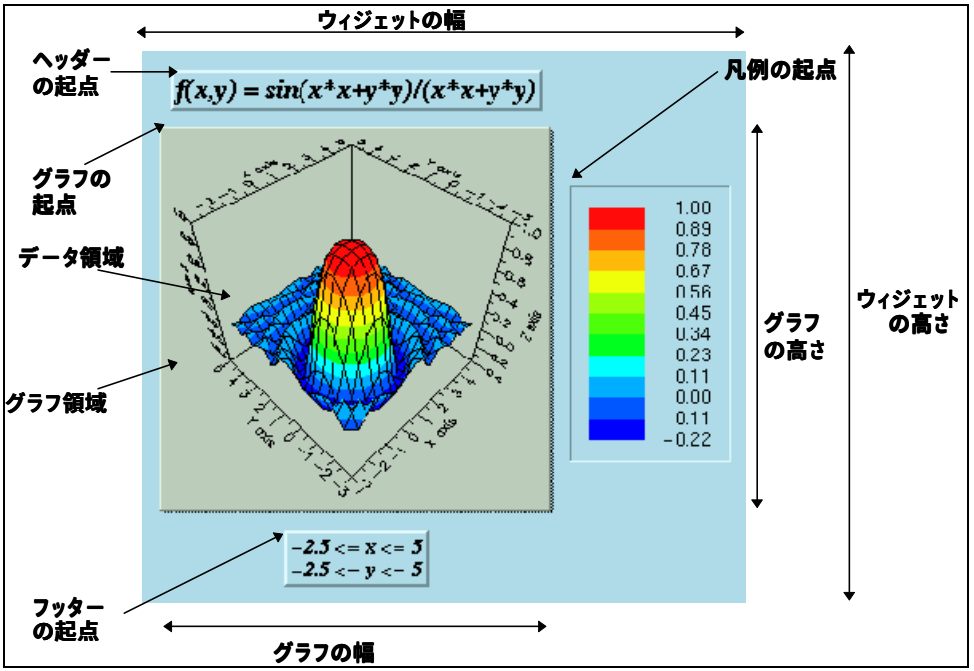


図4　XRT/3dグラフ領域

2.3　浮動小数点リソース

XRT/3dは、全ての浮動小数点のリソース値にdoubleを使用します。このリソースをセットするときは、*Xrt3dDoubleToArgVal()*を使用してください。例えば、現在のY軸の最大値を3倍にするには、次のようにしてください。

```
double ymax;
XtVaGetValues(mygraph, XtNxrt3dYMax, &ymax, NULL);
ymax = ymax * 3.0;
XtVaSetValues(mygraph,
  XtNxrt3dYMax, Xrt3dDoubleToArgVal(ymax),
  NULL);
```

2.4 UseDefaultリソース

多くのXRT/3dリソースは、対応するUseDefaultリソース(例: XtNxrt3dHeaderXとXtNxrt3dHeaderXUseDefault等)を持っています。

UseDefaultリソースは、XRT/3dがリソースのデフォルト値を計算するかどうかを指定するBoolean関数です。

例えば、XtNxrt3dHeaderXUseDefaultがTrueであれば、XRT/3dのウィンドウがリサイズされる度に、XRT/3dはヘッダーの位置のX pixel座標に対するデフォルト値を計算します。Trueでなければ、一番最後に指定されたXtNxrt3dHeaderXの値をヘッダーの位置として使用します。

リソースが明示的に設定されていなかったり、XRT/3dによって計算されていない場合は、対応するUseDefaultをFalseにする設定は無視されます。また、これらのリソースを設定すると、副作用でUseDefaultはFalseにセットされます。

次のコードは、XtNxrt3dXMaxを強制的に現在の値にします。この副作用でXtNxrt3dXMaxUseDefaultはFalseになります。

```
double xmax;
XtVaGetValues(mygraph,
  XtNxrt3dXMax, &xmax, NULL);
XtVaSetValues(mygraph,
  XtNxrt3dXMax, Xrt3dDoubleToArgVal(xmax),
  NULL);
```

次のコードでデフォルトに戻り、XRT/3dはグラフ描画時にXtNxrt3dXMaxの値を計算します。

```
XtVaSetValues(mygraph,
  XtNxrt3dXMaxUseDefault, True,
  NULL);
```

リソース・ファイルでのUseDefaultの設定は無視されます。

2.5 ポインタ・リソース

幾つかのXRT/3dのリソースは、*XtGetValues()*で使用されると、構造体か文字のポインタを返します。このポインタをメモリ内のデータ変更に使用することはできません。ポインタが指すものは、全て"read only"です。

但し、XtNxrt3dSurfaceDataとXtNxrt3dZoneDataのリソースは例外です。これらのリソースは、常にプログラムのメモリにあるデータ構造体を指しています。

XRT/3dは、ポインタ・リソースを通して渡された全てのデータのコピーを作成します。(XtNxrt3dSurfaceDataとXtNxrt3dZoneDataは例外)

"duplicating"(複写)の手続き、*Xrt3dDupContourStyles()*、*Xrt3dDupDistnTable()*、*Xrt3dDupDataStyles()*、*Xrt3dDupStrings()*、*Xrt3dDupValueLabels()*、*Xrt3dDupXYColors()*は、データのコピー作成に使用できます。これらのコピーは、対応する"free"の手続き(*Xrt3dFreeStrings()*等)を使用して解放できます。Appendix Cに手続きのリストがありますので参照してください。

例えば、プログラムで最初のヘッダー・ストリングと最初のフッター・ストリングを交換する場合、次のコードが使用できます。

```
char  **hs, **fs, **myhs, **myfs, *tmp;
XtVaGetValues(graph,       /* read-only ポインタを得る */
  XtNxrt3dHeaderStrings, &hs,
  XtNxrt3dFooterStrings, &fs,
  NULL);

myhs = Xrt3dDupStrings(hs);    /* コピーを作成する */
myfs = Xrt3dDupStrings(fs);

tmp = myfs[0];  /* 一番目を交換する */
myfs[0] = myhs[0];
myhs[0] = tmp;

XtVaSetValues(graph,       /*リセットする*/
  XtNxrt3dHeaderStrings, myhs,
  XtNxrt3dFooterStrings, myfs,
  NULL);

Xrt3dFreeStrings(myhs);    /*コピーを解放する*/
Xrt3dFreeStrings(myfs);
```

2.6 リソース・アップデートのバッチ処理

通常、リソースの変更は、*XtSetValues()*または*XtVaSetValues()*の直後に行われます。再描画される前に、ウィジェットのリソースに幾つかの変更を加えたい場合は、`XtNxrt3dRepaint`を`False`にセットしてください。全てのリソースの変更は、`XtNxrt3dRepaint`が`True`にセットされるまでバッチ処理されます。

2.7 色

ウィジェットや各グラフ領域の背景、前景に色を指定できます。デフォルトでは、グラフ領域の背景色は透明、前景色はウィジェットの前景色に設定されています。

メッシュの上部と下部、サーフェスの影、等高線、ゾーンの塗りつぶし等も、色の指定が可能です。

XRT/3dはデータ表示の際、デフォルトで色を選択しますので、ほとんどのアプリケーションでは色の指定をする必要はありません。

色の指定

色は以下のように、色名か16進数のRGB[赤(R)、緑(G)、青(B)]構成を含むストリングで指定します。

```
char * my_named_color = "LemonChiffon";
char * my_rgb_color = "#228B22";
```

色名は、X Xindowサーバのカラー・データベースから選択してください。通常、このデータベースは、*/usr/lib/X11/rgb.txt*にあります。

RGB値で指定する場合、ストリングは'#'で始め、次にRGB(赤、緑、青)の二桁の16進数値が来ます。上の例の`my_rgb_color`では、赤が0x22、緑が0x8B、青が0x22です[1] 。

モノクロのXサーバの場合、XRT/3dで作成されたアプリケーションは次のように表示されます。

- ゾーンの色やサーフェスの上下の色に、黒と白のピクセルの割合を調節して色の濃淡を表す。
- 黒以外の色は、ウィンドウまたはグラフ領域の背景色に使用されると白に表示される。
- 白以外の色は全て黒に表示される。

一度XRT/3dウィジェットが作成されると、アプリケーションは、カラーマップのアップデートや変更ができません。色の変更は、ウィジェットのリソースで行ってください。(`XtNxrt3dBackgroundColor`、`XtNxrt3dContourStyles`等)

カラーマップのフラッシュ

X Windowサーバのカラーマップは共用リソースです。ある状況においては、全てのアプリケーションで表示中の色の数が、使用可能なカラーマップのエントリー数より多いことがあります。この場合、幾つかのウィンドウでユーザがマウスを動かした時、その色が他の色に変更(フラッシュ)してしまいます。

サーバがスタティックな色だけを提供する場合は、このカラー・フラッシュが起こりません。アプリケーションには、指定通りの色ではありませんが、カラーマップ内の一番近い色が選択されます。

しかし、多くのアプリケーションではダイナミックな色を必要とします。この場合、各アプリケーションは指定通りの色を使用できますが[2] 、サーバの使用可能な色の数やアプリケーションの数と状態によってカラーマップ・フラッシュが起こります。

ダイナミック・カラーマップで実行中、XRT/3dは、アプリケーションの要求に従って自動的にサーバのカラー・セルの割り当てまたは、割り当て解除を行い、XRT/3dアプリケーションによるカラーマップ・リソースへの要求をできる限り少なくします。

---

[1] 色の指定の構文は例の通りでなくてもよく、Xlibの*XParseColor()*がサポートするストリングなら、どれでも使用できます。

[2] ハードウェアの表示能力の範囲内

XRT/3dアプリケーションが使用可能な数以上の色を要求すると、使用不可な色は白か黒で表示されます。

色に影響するXRT/3dのリソースを変更した時にもフラッシュが起こることがあります。色が変更され、XRT/3dが再描画が不必要と判断すると、サーバのカラーマップのルックアップ・エントリー(lookup entry)だけが変更されます。

2.8 リソース・ファイル

リソースの値を設定するには、Cコードより、リソース・ファイルを使用した方がよい場合があります。リソース・ファイルには次の利点があります。

- スピード - コンパイルしないので、変更が瞬時に確認できる。
- 柔軟性 - エンドユーザがニーズにあわせてリソースの値を変更できる。

`UseDefault`リソースの例外を除いて、XRT/3dの全てのリソースはリソース・ファイルで設定ができます。但し、ほとんどのリソースはエンドユーザが変更を加えたりするには複雑すぎるかもしれません。アプリケーションの全てのウィジェットのリソース・ファイルは、組み合わせてアプリケーションのデフォルト・ファイルを作成することができます。

リソース・ファイルとその使用法についての詳細は、Xt Intrinsicsの資料等をご覧ください。Appendix Gには、XRT/3dのサンプルのリソース・ファイルとリソースのフォーマットについての解説があります。

2.9 デバッグ

`XtNxrt3dDebug`を`True`にセットした場合、XRT/3dは誤りを見つけると(*xstep*や*ystep*を負の値に設定した場合等)、標準エラー出力にメッセージをプリントします。

3

XRT/3dのプログラミング

リソース ■ 軸のコントロール

凡例の表示 ■ 遠近法

軸のスケーリング ■ 軸の注釈

グリッド線 ■ ヘッダーとフッターのテキスト

ヘッダー、フッターと凡例のフォント ■ 領域の位置決め

領域の枠線 ■ 前景と背景の色

背景のイメージ ■ ダブルバッファリング ■ 出力と印刷

3.1 リソース

この章では、XRT/3dのリソースについて解説します。これらのリソースは、覚えていただく必要はなく、ほとんどのグラフでは、リソースをデフォルト設定のままにしておくことができます。

Appendix AにはXRT/3dの全てのリソースの一覧がありますので参照してください。リソースの中には、決まったグラフの種類にしか使用できないものもあります。

Resource	Type
XtNxrt3dContourStyles	Xrt3dContourStyle **
XtNxrt3dDataAreaBackgroundColor	char *
XtNxrt3dDataStyles	Xrt3dDataStyles **
XtNxrt3dDataStylesUseDefault	Boolean
XtNxrt3dDistnMethod	Xrt3dDistnMethod
XtNxrt3dDistnTable	Xrt3dDistnTable *
XtNxrt3dDrawDropLines	Boolean
XtNxrt3dDrawMesh	Boolean
XtNxrt3dDrawShaded	Boolean
XtNxrt3dDrawContours	Boolean
XtNxrt3dDrawZones	Boolean
XtNxrt3dDrawHiddenLines	Boolean
XtNxrt3dImage	Xtr3dImage
XtNxrt3dImageLayout	Xrt3dImageLayout
XtNxrt3dMeshBottomColor	char *
XtNxrt3dMeshTopColor	char *
XtNxrt3dNumDistnLevels	int
XtNxrt3dPerspectiveDepth	double
XtNxrt3dProjectZMax	int
XtNxrt3dProjectZMin	int
XtNxrt3dSolidSurface	Boolean
XtNxrt3dSurfaceBottomColor	char *
XtNxrt3dSurfaceTopColor	char *
XtNxrt3dSurfaceData	Xrt3dData *
XtNxrt3dTextList	WidgetList
XtNxrt3dType	Xrt3dType
XtNxrt3dViewNormalized	Boolean
XtNxrt3dViewScale	double
XtNxrt3dView[XY]Translate	double
XtNxrt3d[XY]MeshFilter	int
XtNxrt3d[XY]MeshShow	Boolean
XtNxrt3d[XYZ]Rotation	double
XtNxrt3d[XYZ]Scale	double
XtNxrt3dZoneData	Xrt3dData *
XtNxrt3dZoneMethod	Xrt3dZoneMethod

図5　データ表示のリソース

Resource	Type
XtNxrt3dGraphBackgroundColor	char *
XtNxrt3dGraphBorder	Xrt3dBorder
XtNxrt3dGraphBorderWidth	Dimension
XtNxrt3dGraphForegroundColor	char *
XtNxrt3dGraphHeight	Dimension
XtNxrt3dGraphHeightUseDefault	Boolean
XtNxrt3dGraphImage	Xrt3dImage
XtNxrt3dGraphImageLayout	Xrt3dImageLayout
XtNxrt3dGraphWidth	Dimension
XtNxrt3dGraphWidthUseDefault	Boolean
XtNxrt3dGraph[XY]	Position
XtNxrt3dGraph[XY]UseDefault	Boolean

図6　グラフのリソース

Resource	Type
XtNxrt3dAxisFont	XmFontList
XtNxrt3dAxisTitleFont	XmFontList
XtNxrt3dAxisStrokeFont	Xrt3dStrokeFont
XtNxrt3dAxisStrokeSize	Dimension
XtNxrt3dAxisTitleStrokeFont	Xrt3dStrokeFont
XtNxrt3dAxisTitleStrokeSize	Dimension
XtNxrt3d[XYZ]GridLineStyle	Xrt3dLineStyle
XtNxrt3d[XYZ]GridLineStyleUseDefault	Boolean
XtNxrt3d[XY]DataLabels	char **
XtNxrt3d[XYZ]AnnoMethod	Xrt3dAnnoMethod
XtNxrt3d[XYZ]AxisShow	Boolean
XtNxrt3d[XYZ]AxisTitle	char *
XtNxrt3d[XYZ]GridLines	int
XtNxrt3d[XYZ]Max	double
XtNxrt3d[XYZ]MaxUseDefault	Boolean
XtNxrt3d[XYZ]Min	double
XtNxrt3d[XYZ]MinUseDefault	Boolean
XtNxrt3d[XYZ]ValueLabels	Xrt3dValueLabel **
XtNxrt3dZOrigin	double

図7　軸のリソース

Resource	Type
XtNxrt3dHeaderAdjust	Xrt3dAdjust
XtNxrt3dHeaderBackgroundColor	char *
XtNxrt3dHeaderBorder	Xrt3dBorder
XtNxrt3dHeaderBorderWidth	Dimension
XtNxrt3dHeaderForegroundColor	char *
XtNxrt3dHeaderImage	Xrt3dImage
XtNxrt3dHeaderImageLayout	Xrt3dImageLayout
XtNxrt3dHeaderImageMinimumSize	Boolean
XtNxrt3dHeaderStrings	char **
XtNxrt3dHeaderFont	XmFontList
XtNxrt3dHeaderHeight	Dimension
XtNxrt3dHeaderWidth	Dimension
XtNxrt3dHeader[XY]	Position
XtNxrt3dHeader[XY]UseDefault	Boolean
XtNxrt3dFooterAdjust	Xrt3dAdjust
XtNxrt3dFooterBackgroundColor	char *
XtNxrt3dFooterBorder	Xrt3dBorder
XtNxrt3dFooterBorderWidth	Dimension
XtNxrt3dFooterFont	XmFontList
XtNxrt3dFooterForegroundColor	char *
XtNxrt3dFooterHeight	Dimension
XtNxrt3dFooterImage	Xrt3dImage
XtNxrt3dFooterImageLayout	Xrt3dImageLayout
XtNxrt3dFooterImageMinimumSize	Boolean
XtNxrt3dFooterStrings	char **
XtNxrt3dFooterWidth	Dimension
XtNxrt3dFooter[XY]	Position
XtNxrt3dFooter[XY]UseDefault	Boolean

図8　ヘッダーとフッターのリソース

Resource	Type
XtNxrt3d[XY]BarFormat	Xrt3dBarFormat
XtNxrt3d[XY]BarSpacing	double
XtNxrt3dXYColors	Xrt3dXYColor **

図9　棒のリソース

Resource	Type
XtNxrt3dExposeCallback	XtCallbackList
XtNxrt3dMapCallback	XtCallbackList
XtNxrt3dModifyCallback	XtCallbackList
XtNxrt3dPickCallback	XtCallbackList
XtNxrt3dResizeCallback	XtCallbackList
XtNxrt3dRotateCallback	XtCallbackList
XtNxrt3dTransformCallback	XtCallbackList

図10　コールバックのリソース

Resource	Type
XtNxrt3dLegendAnchor	Xrt3dAnchor
XtNxrt3dLegendBackgroundColor	char *
XtNxrt3dLegendBorder	Xrt3dBorder
XtNxrt3dLegendBorderWidth	Dimension
XtNxrt3dLegendDistnRange	Xrt3dDistnRange
XtNxrt3dLegendFont	XmFontList
XtNxrt3dLegendForegroundColor	char *
XtNxrt3dLegendHeight	Dimension
XtNxrt3dLegendImage	Xrt3dImage
XtNxrt3dLegendImageLayout	Xrt3dImageLayout
XtNxrt3dLegendLabelFilter	int
XtNxrt3dLegendLabelFunc	Function
XtNxrt3dLegendOrientation	Xrt3dAlign
XtNxrt3dLegendShow	Boolean
XtNxrt3dLegendStrings	char **
XtNxrt3dLegendStyle	Xrt3dLegendStyle
XtNxrt3dLegendTitle	char *
XtNxrt3dLegendWidth	Dimension
XtNxrt3dLegend[XY]	Position
XtNxrt3dLegend[XY]UseDefault	Boolean

図11　凡例のリソース

Resource	Type
XtNxrt3dBackgroundColor	char *
XtNxrt3dColorFunc	XtFunction
XtNxrt3dDebug	Boolean
XtNxrt3dDoubleBuffer	Boolean
XtNxrt3dForegroundColor	char *
XtNxrt3dPreviewMethod	Xrt3dPreviewMethod
XtNxrt3dRepaint	Boolean

図12　その他のリソース

Resource	Type
XtNxrt3dTextAdjust	Xrt3dAdjust
XtNxrt3dTextAttachIndex[XY]	int
XtNxrt3dTextAttachIndexPoint	int
XtNxrt3dTextAttachIndexSeries	int
XtNxrt3dTextAttachMethod	Xrt3dTextAttachMethod
XtNxrt3dTextAttachPixel[XY]	Position
XtNxrt3dTextAttachPoint[XYZ]	double
XtNxrt3dTextBackgroundColor	char *
XtNxrt3dTextBorder	Xrt3dBorder
XtNxrt3dTextBorderWidth	Dimension
XtNxrt3dTextFont	XmFontList
XtNxrt3dTextForegroundColor	char *
XtNxrt3dTextImage	Xrt3dImage
XtNxrt3dTextImageLayout	Xrt3dImageLayout
XtNxrt3dTextImageMinimumSize	Boolean
XtNxrt3dTextLineShow	Boolean
XtNxrt3dTextOffset[XY]	Position
XtNxrt3dTextPlane	int
XtNxrt3dTextPrintFont	char *
XtNxrt3dTextShow	Boolean
XtNxrt3dTextStrings	char **
XtNxrt3dTextStrokeFont	Xrt3dStrokeFont
XtNxrt3dTextStrokeSize	Dimension

図13　テキスト・オブジェクトのリソース

3.2 軸のコントロール

図7にリストアップされたリソースは、軸の表示をコントロールします。

軸の表示

XtNxrt3d[XYZ]AxisShowは、XRT/3dに軸を表示するかどうかを指示します。Falseにセットされると、軸は表示されません。

注釈のフォント

軸の注釈は、XtNxrt3dAxisStrokeFontで指定されるフォントを使用します[1]。デフォルトはXRT3D_SF_ROMAN_SIMPLEXフォントです。その他のフォントは、Appendix Eに一覧がありますので参照してください。

軸のフォントのサイズは、ユニット・キューブの長さの1/1000単位で測ります。デフォルトの軸のフォント・サイズは80で、この文字はユニット・キューブの高さの8%の大きさです。(ユニット・キューブの定義は、2.2にあります。)

タイトル

XtNxrt3d[XYZ]Titleは各軸のタイトルに使用されます。タイトルのフォントはXtNxrt3d[XYZ]AxisTitleStrokeFont、サイズはXtNxrt3d[XYZ]AxisTitleStrokeSizeで指定されます。

最小値と最大値

XtNxrt3d[XYZ]MaxとXtNxrt3d[XYZ]Minは、各軸の最小値と最大値を指定します。XRT/3dは、デフォルトで、実際のデータの最大値と最小値に基づいて軸の範囲を設定し、データが変更されると、軸を自動的にアップデートします。データの最小値と最大値が、グリッドのインクレメントの間にセットされた場合、一番近くの高い方のグリッド・ポイントに合うように、値は自動的に大きくなります。

表示するデータの範囲は、これらのリソースを使用して設定することができます。アプリケーションによってセットされている場合は、データが変更しても軸の範囲はアップデートされません。デフォルトに戻すには、XtNxrt3d[XYZ]MaxUseDefaultまたはXtNxrt3d[XYZ]MinUseDefaultをTrueにしてください。

(注意) Z軸の最小と最大は、データのZ範囲の内側には指定できません。XtNxrt3dZMin/XtNxrt3dZMaxが設定されている場合、これらをZ範囲の内側にしてしまうデータ変更が加えられると、XRT/3dは対応するUseDefaultリソースをTrueにセットします。

3.3 凡例の表示

XRT/3dは、等高線やゾーン、ポイントが描かれる時に自動的に凡例を作成します。凡例が必要ない時は、XtNxrt3dLegendShowをFalseにしてください。ゾーンが描画されずに等高線だけが描かれる場合は、等高線のリストが凡例として作成されます。ゾーンが描画されれば、各レベルの塗りつぶし色が凡例にリストアップされます。ポイントが描画される場合は、そのシンボルが凡例に表示されます。

[1] 軸の注釈には、日本語は使用できません。

デフォルトで、XRT/3dは凡例の内容を縦に、グラフ領域の右側(東)に配置します。

方向

凡例の方向は、XtNxrt3dLegendOrientationでコントロールします。このリソースはXRT3D_ALIGN_VERTICALまたはXRT3D_ALIGN_HORIZONTALのどちらかにセットします。凡例が大きすぎて一行または一列に収まらない場合は、複数行または複数列に自動的にレイアウトされます。

アンカー

凡例の位置はグラフ領域の位置に相対し、XtNxrt3dLegendAnchorによってコントロールできます。有効な値は方位磁石の8ポイントに対応しています。
(XRT_3D_ANCHOR_NORTH, XRT3D_ANCHOR_SOUTH, XRT_3D_ANCHOR_EAST, XRT3D_ANCHOR_WEST, XRT3D_ANCHOR_NORTHWEST, XRT3D_ANCHOR_NORTHEAST, XRT3D_ANCHOR_SOUTHEAST, XRT3D_ANCHOR_SOUTHWEST)

スタイル

XtNxrt3dLegendStyleはXRT3D_LEGEND_STYLE_CONTINUOUS (デフォルト)またはXRT3D_LEGEND_STYLE_STEPPEDのどちらかに設定できます。図14は、これらのスタイルの例です。どちらも同じ情報を表しますが、Continuousの方がスペースを取りません。

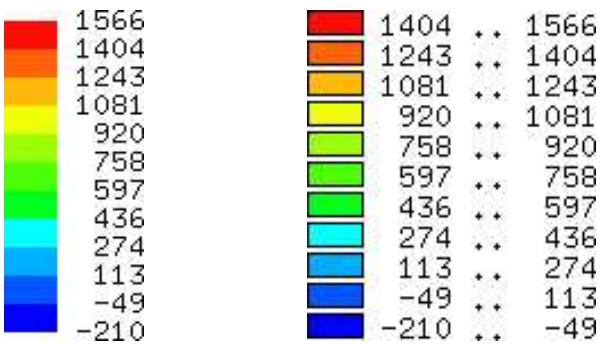


図14　ContinuousとSteppedの凡例

凡例のオプション

凡例のラベルは、XtNxrt3dLegendLabelFilterによって減らしたり、欠けた範囲を表すように間隔を変更したりすることができます。このリソースが*n*にセットされると、*n*番目毎の凡例が表示されます。

凡例の位置と枠線

凡例の位置の指定は3.10、枠線の指定については3.11を参照してください。

凡例のタイトル

XtNxrt3dLegendTitleを使用して凡例の中央上に一行のテキストをタイトルとして表示することができます。

任意の凡例

凡例のテキストを任意に設定することができます。詳細は、8.5に解説されています。

3.4 遠近法

XtNxrt3dPerspectiveDepthは、ユニット・キューブをスクリーンに表示する時の遠近法をコントロールします。大きい値に比べ、小さい値の法が遠近法の効果がよく表れます。有効な値は1とXRT3D_HUGE_VALの間です。スキャッター・プロットでも遠近法をセットすることができますが、効果はありません。

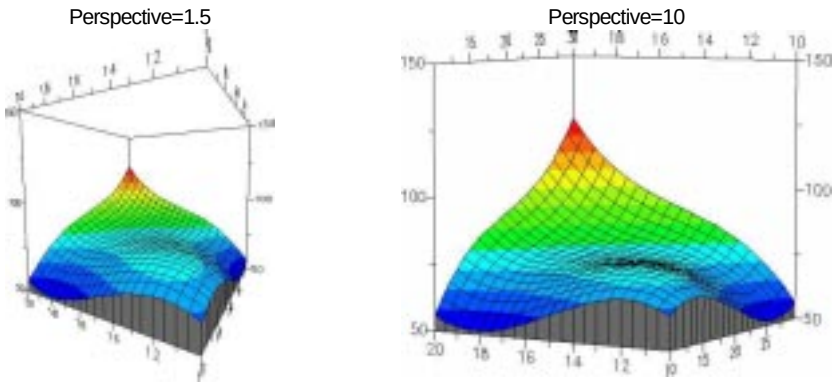


図15　遠近法

3.5 軸のスケーリング

軸のスケーリングのリソース、XtNxrt3d[XYZ]Scaleは、ユニット・キューブのX、Y、Z次元をコントロールします。3D表示をXに対してY方向を二倍にする場合、Yのスケールを Xのスケールの二倍にします。Zスケールを高低に調節すると、サーフェスのイメージが上下に伸び縮みします。

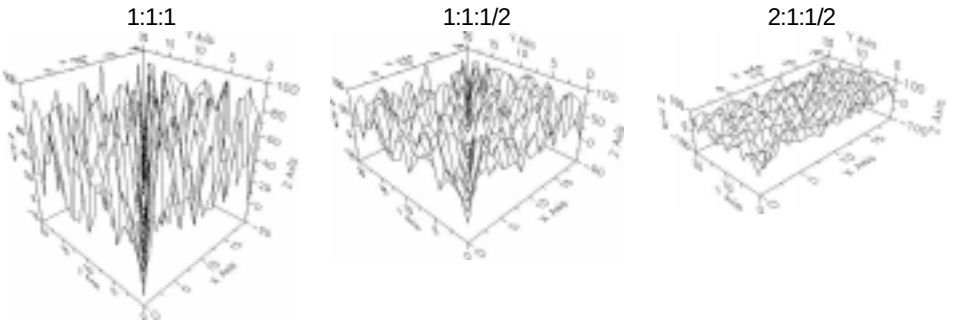


図16　スケーリングの例 (X:Y:Z)

3.6　軸の注釈

X軸、Y軸には3通り、Z軸には2通りの軸の注釈法があります。軸の注釈法はXtNxrt3d[XYZ]AnnoMethodを用いて設定でき、次の値の一つを取ります。

```
typedef enum {
  XRT3D_ANNO_VALUE,
  XRT3D_ANNO_DATA_LABELS,  /*X, Y軸のみ */
  XRT3D_ANNO_VALUE_LABELS
} Xrt3dAnnoMethod;
```

値(Value)
XtNxrt3d[XYZ]AnnoMethodがXRT3D_ANNO_VALUEである時、XRT/3dは自動的にデータの範囲に基づいて軸に注釈を付けます。これは、Z軸と、XtNxrt3dTypeがXRT3D_TYPE_SURFACEの時のX軸、Y軸に適しています。これはデフォルトの注釈法です。

データ・ラベル(Data Label)
サーフェスの線、棒の行や列にラベルを付ける時は、XRT3D_ANNO_DATA_LABELSを使用します。この方法は、XtNxrt3d[XY]DataLabelsで提供されたストリングのリストを使用します。各ラベルは*Xrt3dSetNthDataLabel()*で提供され、*Xrt3dGetNthDataLabel()*で取り出すことができます。この方法は、ポイント・データで使用された時は、ラベルを表示しません。

値ラベル(Value Label)
XtNxrt3d[XYZ]AnnoMethodの値をXRT3D_ANNO_VALUE_LABELSに設定することにより、軸の任意の位置にラベルを付けることができます。

位置はXtNxrt3d[XYZ]ValueLabelsで指定でき、これは次の構造体のポインタの配列を取ります。

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

軸の範囲外のラベルは表示されません。各値ラベルは*Xrt3dSetValueLabel()*で追加/削除したり、*Xrt3dGetValueLabel()*で取り出すことができます。

3.7 グリッド線

グリッド線はXtNxrt3d[XYZ]GridLinesを用いて、各三つの基本plane(面)に表示することができます。これらのリソースは整数を値として取り、次の定数から組み合わされます。

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

例えば、全てのplaneにXグリッド線を引く場合、XtNxrt3dXGridLinesを値(XRT3D_XY_PLANE | XRT3D_XZ_PLANE)にセットします。グリッド線を削除するには、この値を0にします。2Dグラフでは、グリッド線はXtNxrt3d[XY]GridLinesのリソースが0以外であれば、いつでも表示されます。

グリッド線は、(軸の注釈法に関係なく)軸の注釈のある箇所に表示されます。

グリッド線の色、幅、線スタイルはXtNxrt3d[XYZ]GridLineStyleでコントロールできます。デフォルトのグリッド線は、1ピクセル幅の実線で、XtNxrt3dGraphForegroundColorを使用し、値がNULLであれば、XtNxrt3dForegroundColorを使用します。XtNxrt3d[XYZ]GridLineStyleUseDefaultでデフォルトの設定に戻ります。

3.8 ヘッダーとフッターのテキスト

ヘッダーとフッターは複数行のテキストを表示できます。テキストは、XtNxrt3dHeaderAdjustまたはXtNxrt3dFooterAdjustの値により、行揃えされ、XRT3D_ADJUST_LEFT、XRT3D_ADJUST_RIGHT、XRT3D_ADJUST_CENTERは、それぞれテキストを左、右、中央に揃えます。デフォルトはXRT3D_ADJUST_CENTERです。

ヘッダーとフッターのテキストは、XtNxrt3dHeaderStringsとXtNxrt3dFooterStringsを用いて指定します。どちらのリソースもストリングのNULL終端配列を値として持ちます。

次のコードは、二行のテキストを左揃えにヘッダーに表示します。

```
static char *hs[] = {"Analysis Results",
     "Sample Group A", NULL}
XtVaSetValues(my_graph,
  XtNxrt3dHeaderStrings, hs,
  XtNxrt3dHeaderAdjust, XRT3D_ADJUST_LEFT,
  NULL);
```

3.9 ヘッダー、フッターと凡例のフォント

ヘッダー、フッター、凡例には、X Windowフォントが指定できます。

フォント名で指定する場合は、(X Windowプログラム)*xlsfonts*または(Xlibの手続き)*XListFonts()*を使用して、サーバでどのフォントが使用可能か確認することができます。フォント・ストリングからXmFontListを取得する場合は、Xtの手続き、*XtConvertAndStore()*を使用してください。尚、フォントの検索と設定についての詳細は、Xtに関する資料を参照してください。

```
static XmFontList
CvtStringToFontList(parent, font_name)
Widget   parent;
char     *font_name;
{
    XrmValue  to;
    XrmValue  from;

    from.addr = font_name;
    from.size = strlen(font_name);
    to.addr = NULL;
    to.size = 0;

    if (!XtConvertAndStore(parent, XmRString, &from, XmRFontList,&to)) {
         fprintf(stderr, "Conversion of %s to XmFontList failed!\n",font_name);
         return((XmFontList)0);
    }
    return(*(XmFontList *)to.addr);
}


XmFontList hf, ff;
hf=CvtStringToFontList(graph, "-*-times-medium-*-*--18-*-iso8859-*");
ff=CvtStringToFontList(graph, "9x15");
if (hf != NULL && ff != NULL) {
    XtVaSetValues(graph,
         XtNxrt3dHeaderFont, hf,
         XtNxrt3dFooterFont, ff,
         NULL);
}
```

図17　XmFontListリソースの設定

3.10 領域の位置決め

各四つの領域(ヘッダー、フッター、凡例、グラフ)は、実行時にXRT/3dによって配置されます。デフォルトの位置は以下を始め、あらゆる要因に依存します。

- ウィンドウの現在の幅と高さ。
- 凡例、ヘッダー、フッターのサイズ。(これらは、使用されるテキストやフォントに依存する。)
- `XtNxrt3dLegendAnchor`の値。
- ユーザ・プログラムによって決定されている位置。

XRT/3dのデフォルトの位置決めアルゴリズムは、ヘッダー、フッター、凡例のサイズと位置を最初に決め、グラフを残りの矩形の領域に入るようサイズを変更し、配置します。

位置決めリソースを使用して、各領域の位置を決定することができます。

`XtNxrt3dHeaderX`	`XtNxrt3dHeaderY`	`(XtNxrt3dHeaderWidth)`	`(XtNxrt3dHeaderHeight)`
`XtNxrt3dFooterX`	`XtNxrt3dFooterY`	`(XtNxrt3dFooterWidth)`	`(XtNxrt3dFooterHeight)`
`XtNxrt3dLegendX`	`XtNxrt3dLegendY`	`(XtNxrt3dLegendWidth)`	`(XtNxrt3dLegendHeight)`
`XtNxrt3dGraphX`	`XtNxrt3dGraphY`	`XtNxrt3dGraphWidth`	`XtNxrt3dGraphHeight`

(**括弧内のリソースは、**read-only**です。**)

図18　位置決めのリソース

*XtGetValues()*で使用されると、位置決めリソースは、ウィジェットが最後に表示された時に使用された値を返します。幅と高さのリソースの型は`Dimension`で、X、Yのリソースは`Position`です。`Dimension`と`Position`はXt Intrinsicsで定義されます。

プログラムでは、ウィンドウのサイズが変更された時、再計算の準備ができるまで位置決めのリソースをセットさせないでください。ウィンドウのリサイズの処理についてはセクション7.7をご覧ください。

ある状況においては、位置決めのリソースを一部または全部設定しておいた方が良い場合があります。例えば、プログラムがリアルタイムに変化するデータを表示する場合、デフォルトの位置はデータが変化する度に変更されることがあります。これには、次のように対処してください。

- 全ての位置決めのリソースをハードコーディングする。この場合ウィンドウの大きさの変更はできない。

または、

- グラフが最初に表示される時、またはウィンドウがリサイズされた後にグラフが表示される時、XRT/3dに全ての領域のデフォルトの位置を計算させる。表示後、その値に位置決めリソースをセットする。

3.11 領域の枠線

各四つの領域(ヘッダー、フッター、凡例、グラフ)には枠線を描くことができます。各領域の枠線には、BorderとBoarderWidthのリソースがあります。(XtNxrt3dGraphBorder、XtNxrt3dGraphBorderWidth等)

Borderのリソースは、XRT3D_BORDER_NONE、XRT3D_BORDER_PLAIN、XRT3D_BORDER_3D_OUT、XRT3D_BORDER_3D_IN、 XRT3D_BORDER_ETCHED_IN、XRT3D_BORDER_ETCHED_OUT、XRT3D_BORDER_FRAME_IN、XRT3D_BORDER_FRAME_OUT、XRT3D_BORDER_BEVEL、 XRT3D_BORDER_SHADOW、に設定できます。枠線の幅は、0から20の間です。

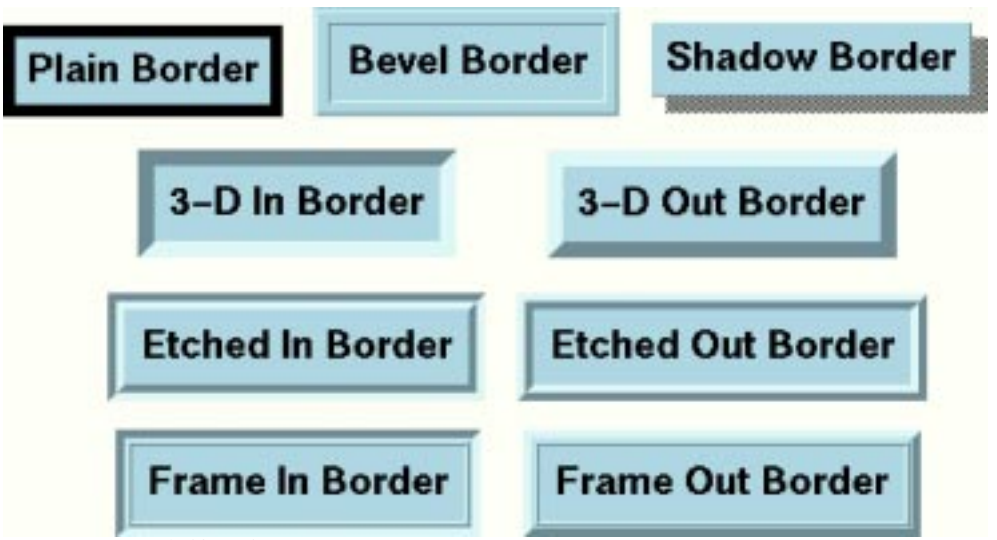


図19　枠線の種類

各領域がそれぞれ交差するように配置されている場合、枠線で交差線が強調されると、見ずらくなることがありますので注意してください。

3.12 前景と背景の色

背景色

ウィンドウの背景色はXtNxrt3dBackgroundColorで指定します。デフォルトは白です。

ヘッダー、フッター、凡例、グラフの各領域は、それぞれ背景色を持ち、デフォルトでNULL(無色)です。例えば、凡例の背景色を変更する時は、XtNxrt3dLegendBackgroundColorリソース、グラフのデータ領域の背景色は、XtNxrt3dDataAreaBackgroundColorで指定できます。

モノクロのサーバでは、黒以外の背景色は白に表示されます。

前景色

ウィンドウの前景色はデフォルトで黒ですが、これは`XtNxrt3dForegroundColor`を使用して指定できます。ヘッダー、フッター、凡例、グラフの各領域の前景にも、他の色が別に指定されていない限り、この前景色が使用されます。例えば、ヘッダーの前景色に他の色を指定する場合は、`XtNxrt3dHeaderForegroundColor`を使用してください。

モノクロのサーバでは、白以外の前景色は黒に表示されます。

一番近い色の取得

XRT/3dは、色のリソースにストリングではなくpixelを使用し、*XAllocColor()*と*XFreeColors()*によって色の管理を行います。カラーマップが足りなくなると、XRT/3dは、白または黒を割り当てますが、`XtNxrt3dColorFunc`リソースを使用することによって、独自の色の割り当て方法を指定することができるようになります。

色の指定には、この新しいリソースの他、(XPMのカラー・マッチングのアルゴリズムを適用した)ファンクションがサンプルとして提供されています。このソースコードは、*$XRTHOME/src/3d/util/x_color_func.c*にあり、*Xrt3dSampleColorFunction()*としてXRT/3dのライブラリにも含まれています。その他、独自の色のファンクションを使用することもできます。

`XtNxrt3dColorFunc`はデフォルトでは`NULL`です。この場合、前のバージョンと同様にbalck/whiteのアルゴリズムを使用します。`NULL`でない場合は、このファンクションを呼び出し、そこで指定する色が使用されます。

独自のファンクションを使用する場合は、必ず下記の条件に従ってください。

1. 使用するファンクションは、次ページのプロトタイプを持つ
2. XRT/3dは、ウィジェットが破棄されたり、ユーザが*XtVaSetValues()*によって色を変更した時には、`XFreeColors`によって色を解放するので、色は(*XAllocColor()*を使用して)必ず割り当てる。色を割り当てることができない場合は、`NO_PIXEL_VALUE`を返すようにする。この場合、ファンクションが指定されていなければ、XRT/3dは黒または白を割り当てる。
3. ファンクションがメモリを割り当てたり、解放したりする必要がある時は、サンプルのファンクションと同じ方法で(do-free flagを使用して)それらを行う。
4. 色のパラメータが`NULL`であれば、色を割り当てずに、`NO_PIXEL_VALUE`を返すようにする。(XRT/3dは、ウィジェットが破棄された時に、color = `NULL`、do-free=`True`でこのファンクションを呼び出す。)

色のファンクションのプロトタイプ

```
unsigned long (*xrt3d_color_func)(
   Display        *display,
   Int            screen,
   Colormap       colormap,
   Xcolor     *color,
   Boolean        do-free
   )
```

背景/前景色の指定

全ての前景と背景色のリソースは、色のストリング(2.7参照)を値として取ります。"LightBlue"や"Wheat"等のパステル・カラーは背景色に、濃い色は前景色に適しています。以下はその例です。

```
XtVaSetValues(my_graph,
 XtNxrt3dBackgroundColor, "LightBlue",
 XtNxrt3dForegroundColor, "NavyBlue",
 XtNxrt3dHeaderBackgroundColor, "LightGray",
 XtNxrt3dFooterForegroundColor, "DarkGreen",
 NULL);
```

グラフのデータ表示に使用されている色を背景に選択しないように注意してください。

3.13 背景のイメージ

ヘッダー、フッター、凡例、グラフ、データ領域の背景にはイメージを表示することができます。これらの各領域に対して、(XtNxrt3dGraphImge、XtNxrt3dGraphImageLayoutのように)ImageとImage Layoutというリソースが用意されています。イメージは、ウィジェット全体に対して表示させることもできます。

ImageリソースはXrt3dImage型で、この型はXrtPDSImage型(9.1参照)と同じです。

次の例では、XRT/3dの変換プログラムを使用してフッターの背景にイメージを表示しています。

```
#include <Xm/XrtPDSImageObj.h>

void
set_image(Widget graph, char *file_name)
{
  Xrt3dImage image;
  XrmValue from, to;

  from.addr = file_name;
  from.size = strlen(file_name) + 1;
  to.addr = (caddr_t)&image;
  to.size = sizeof(image);

if (!XtConvertAndStore(graph, XtRString, &from,
  XtRXrt3dImage, &to)) {
  fprintf(stderr, "Conversion of %s to XtRXrt3dImage failed\n",
      file_name);
  image = NULL;
}
XtVaSetValues(graph,
  XtNxrt3dFooterImage, image,
  NULL);
XrtPDSImageDestroy((XrtPDSImage)image);
}
```

イメージはウィジェットによってコピーされますので、変換プログラムによって作成されたイメージは、必ず破棄してください。イメージを取り去るには、対応するイメージのリソースをNULLにセットしてください。

イメージは、*Xrt3dSetPropString()*を使用してプロパティを設定することにより、セットすることもできます。

```
Xrt3dSetPropString(graph, XRT3D_FOOTER_IMAGE,
      "$XRTHOME/src/icons/shading.xpm");
```

この機能は、*XtConvertAndStore()*ではなく、XRT/3dの変換プログラムを直接に使用し、ウィジェットにイメージをセットする処理は、全て内部で行われます。また、(イメージがリソース・ファイルで指定されように)$XRTHOME環境変数が解釈されます。

サポートされているイメージのフォーマットや、任意のフォーマットを使用する方法に関しては、9.1を参照してください。

Image Layoutリソースは、背景のイメージのサイズや位置を指定します。次の値は、設定できる配置です。

- XRT3D_IMAGELAYOUT_CENTEREDは、領域の中心にイメージをそのまま配置
- XRT3D_IMAGELAYOUT_TILEDは、イメージをタイルのように配置して領域全体を占めるようにする

領域に指定されるイメージは、そこに指定されている背景色より優先されます。但し、ヘッダーに背景色が指定されている場合は、色がイメージを上書きします。

イメージの最小サイズ

背景のイメージは、テキスト領域やヘッダー、フッターにテキストの代わりに使用することができます。XtNxrt3dTextImageMinimumSize、XtNxrt3dHeaderImageMinimumSize、XtNxrt3dFooterImageMinimumSizeのリソースがTrueにセットされると、イメージと同じサイズのラベルが作成されます。

この機能を使用して、例えば、XtNxrt3dTextImageMinimumSizeをテキスト領域に使用し、カスタマイズしたポイントをグラフの任意の場所に配置することができます。

3.14 ダブル･バッファリング

ダブル･バッファリングは、グラフが変化する時のちらつきを減少させます。

XtNxrt3dDoubleBufferがTrueの場合、グラフが変化する時にXRT/3dは次のような働きをします。

- サーバにPixmapを割り当てる(または消去する)
- 完成されたグラフをPixmapに書く
- サーバにPixmapを画面にコピーするよう命令する

XtNxrt3dDoubleBufferがFalseの時は、XRT/3dはグラフが変化する度に、画面のイメージを消去し(ちらつきが起こります)、そして完成したイメージをウィンドウに表示します(グラフが一部分づつ表示されるように見えます)。

デフォルトではXtNxrt3dDoubleBufferはTrueですが、Falseに設定するとグラフのパフォーマンスが向上し、サーバに要求するメモリが減少します。

3.15 出力と印刷

多くのアプリケーションでは、ユーザにグラフのハードコピーを提供する必要があります。XRT/3dは、グラフをファイルやプリンタに出力する機能を備えています。XRT/3dの印刷方法は、グラフのウィンドウが他のウィンドウで隠されていても正常に働きます。各方法は下記に解説されていますが、詳細はAppendix Cにありますので参照してください。

Xrt3dDrawPS()

グラフのPostScript記述を作成します。このルーチンは、ネイティブのPostScriptオペレータを使用してEncapsulated PostScriptファイルを作成します。フォントの選択、色の操作、グラフのサイズや位置決めのパラメータがあります。

Xrt3dOutputPS()

*Xrt3dDrawPS()*と同様に、グラフのEncapsulated PostScript記述を作成しますが、ネイティブのPostScriptオペレータを使用しません。その代わりに、*image*または*colorimage*オペレータを使用し、画面のイメージと同等のPostScript bitmapを作成します。このプリント・ファイルはサイズが大きく、プリンタの解像度を最大限に利用することができないことがあります。

Xrt3dDrawCGM()

*Xrt3dDrawCGM()*は、グラフのComputer Graphics Metafile(CGM)フォーマットを作成します。

Xrt3dOutputXwd()

*Xrt3dOutputXwd()*は、グラフのX Window Dumpフォーマットを作成します。

Xrt3dGetImage()

*Xrt3dGetImage()*は、グラフのXImageのポインタを返します。特別な出力フォーマットを実行したい場合に使用できます。

その他のイメージ・フォーマット

XRT/3dは、ユーザによって登録されたフォーマットのグラフを出力することができます。*Xrt3dSaveImageGeneric()*、*Xrt3dSaveImageJpeg()*、*Xrt3dSaveImagePng()*などのファンクションを使用することができます。

印刷ポップアップ・ウィンドウ

*Xrt3dDrawPS()*の使用法とファイルの*lp*へのパイプの方法を紹介する、印刷のポップアップ・ウィンドウのソースコードが、*$XRTHOME/src/tools/pdbox_3d.c*にありますので参照してください。図20は、このコードから作成されたウィンドウです。

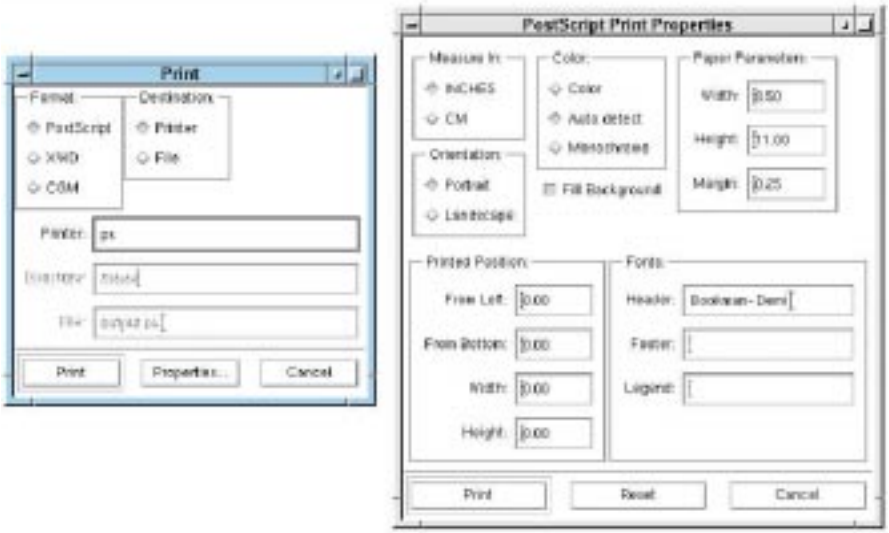


図20　サンプルのコードで作成された印刷のポップアップ･ウィンドウ

4

XRT/3dのプログラミング: サーフェスと棒

サーフェスと棒グラフの15の基本タイプ ■ 棒グラフとヒストグラム
等高線とゾーンの表示 ■ メッシュのコントロール
サーフェスの色 ■ サーフェスの塗りつぶし

4.1 サーフェスと棒グラフの15の基本タイプ

XtNxrt3dSurfaceDataを使用してウィジェットに渡されたグリッド・データ(XRT_DATA_GRIDまたはXRT_DATA_IRGRID型)は、サーフェスか棒グラフに表示されます。グラフのタイプは、XtNxrt3dType(XRT3D_TYPE_SURFACEまたはXRT3D_TYPE_BARまたはXRT3d_TYPE_SCATTER)リソースに依存します。

XRT3D_TYPE_SURFACEかXRT3D_TYPE_BARでは、XRT3D_DATA_POINT型のデータ・セットはグラフ化できません。この型のデータは、スキャッター・プロットにのみ使用されます。

XRT/3dの四つの基本的な表示リソースはDrawMesh、DrawShaded、DrawContours、DrawZonesで、これらを組み合わせて15種類の基本的なサーフェスと棒のグラフが作成できます。[1]

DrawMesh

サーフェス: DrawMeshがTrueの時、XRT/3dはX、YグリッドにZ軸の方向を加えて3Dサーフェスを表示します。このグラフは回転、遠近法のコントロールができます。メッシュの調節は、XtNxrt3dDrawHiddenLines、XtNxrt3d[XY]MeshShow、XtNxrt3d[XY]MeshFilterのリソースを使用し、XtNxrt3dMeshBottomColorとXtNxrt3dMeshTopColorは、色のコントロールに使用できます。

[1] これらのリソースが全てFalseに設定されているとグラフは表示されません。また、これらの四つの表示方法はスキャッター・プロットには関係ありません。

棒: DrawMeshがTrueの場合、XRT/3dは棒の輪郭を描きます。XtNxrt3dZOriginと同じかそれより大きい値の棒はXtNxrt3dMeshTopColorを使用して描かれ、XtNxrt3dZOriginより小さい値にはXtNxrt3dMeshBottomColorが使用されます。XtNxrt3dDrawHiddenLinesがTrueの場合は、棒の線は全て表示されます。

DrawShaded

サーフェス: DrawShadedがTrueの場合、XRT/3dはデータをフラットな(濃淡のない)影付きの3Dサーフェスに表示します。このグラフは回転、透視図法のコントロールができます。色は、XtNxrt3dSurfaceBottomColorとXtNxrt3dSurfaceTopColorで調節できます。

棒: DrawShadedがTrueの場合、XRT/3dは塗りつぶされた棒を表示します。XtNxrt3dZOriginと同じかそれより大きい値の棒はXtNxrt3dSurfaceTopColorを使用して描かれ、XtNxrt3dZOriginより小さい値にはXtNxrt3dSurfaceBottomColorが使用されます。

DrawContours

サーフェス: DrawContoursがTrueの場合、XRT/3dは、(XtNxrt3dDistnMethodまたはXtNxrt3dDistnTableを使用して)データの分布(distribution)を調べ、各分布のレベルに従って等高線を描きます。等高線のスタイル、幅、色は、XtNxrt3dContourStylesで調節できます。

棒: DrawContoursがTrueの場合、XRT/3dは、(XtNxrt3dDistnMethodまたはXtNxrt3dDistnTableを使用して)データの分布を調べ、各分布のレベルに従って棒の周囲に等高線を描きます。等高線のスタイル、幅、色は、XtNxrt3dContourStylesで調節できます。

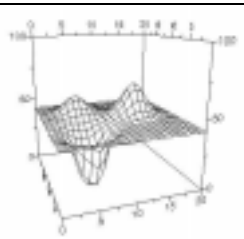
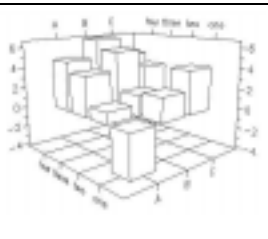
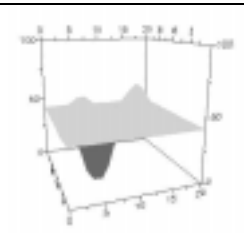
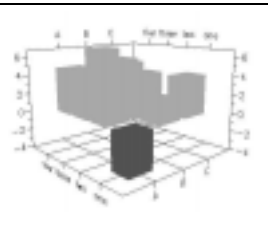
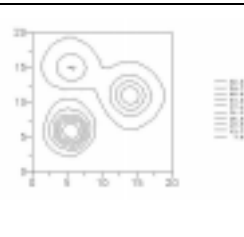
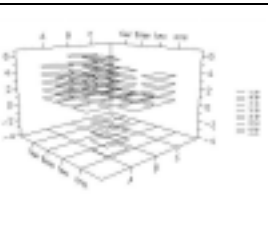
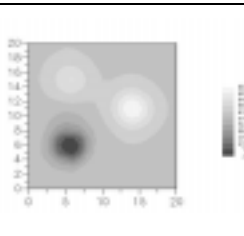
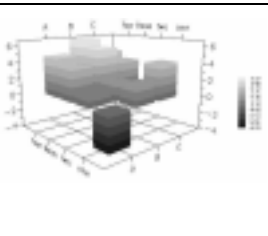
DrawZones

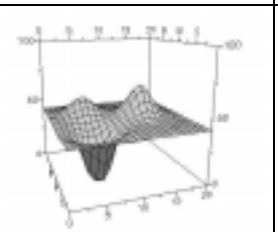
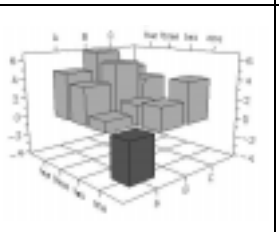
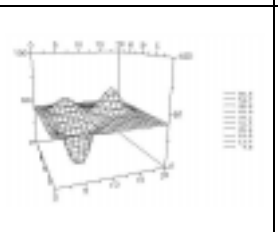
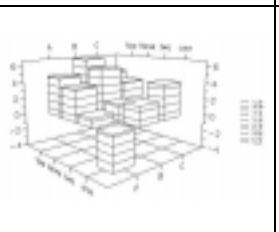
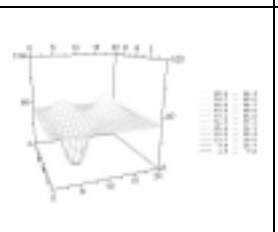
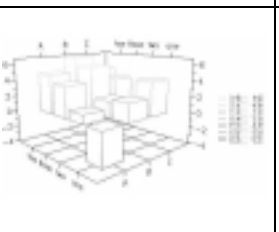
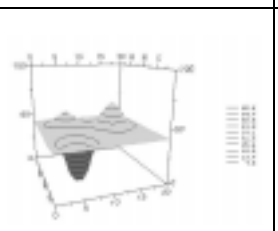
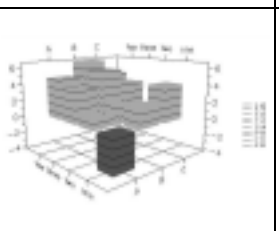
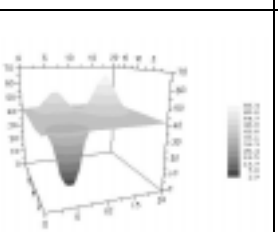
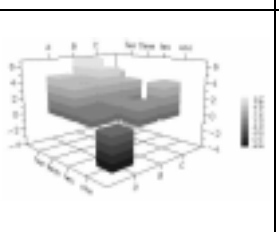
サーフェス: DrawZonesがTrueの場合、XRT/3dは、(XtNxrt3dDistnMethodまたはXtNxrt3dDistnTableを使用して)データの分布(distribution)を調べ、各分布のレベルを色付けします[1]。各レベルの色はXtNxrt3dContourStylesで指定できます。

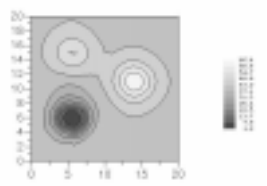
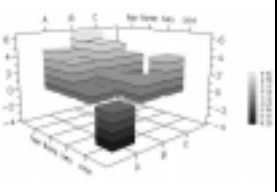
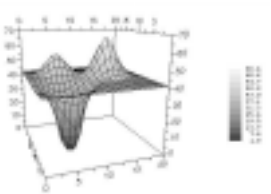
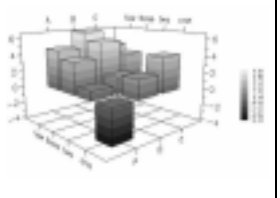
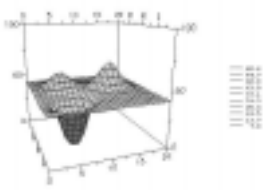
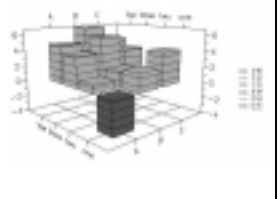
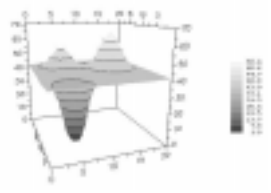
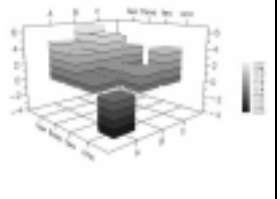
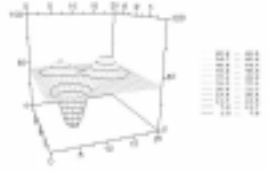
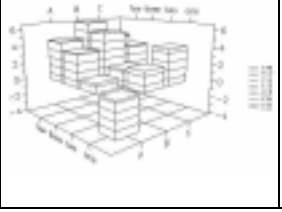
棒: DrawZonesがTrueの場合、XRT/3dは、(XtNxrt3dDistnMethodまたはXtNxrt3dDistnTableを使用して)データの分布を調べ、各棒の分布のレベルをそれぞれ色付けします[1]。色はXtNxrt3dContourStylesで調節できます。XtNxrt3dZoneDataが提供されている場合は、各棒が(一本毎に一色で)塗りつぶされますが、提供されていない場合は、棒は分布表(distribution table)に従って(一本が分布レベルに従って)色分けされます。

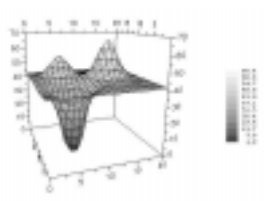
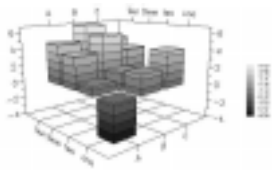
[1] DrawMeshがTrueでDrawShadedがFalseであると、塗りつぶしの色は各レベルのメッシュ線に使用されます。

次の表は、15種類の基本的なグラフの一覧です。

Graph Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface	Bar	Comments
1	T	F	F	F			**Mesh** サーフェスをメッシュ、棒を輪郭で表示。
2	F	T	F	F			**Shaded** サーフェスと棒をフラットな(濃淡のない)影で表示。(原点を堺にして)上部と下部の色は、それぞれセットできる。
3	F	F	T	F			**Contours** 等高線が分布(distribution)レベルを表示。
4	F	F	F	T			**Zones*** 塗りつぶし色が分布(distribution)レベルを表示。

Graph Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface	Bar	Comments
5	T	T	F	F			**Mesh, Shaded** サーフェスをメッシュ、棒を輪郭で表示。フラットな(濃淡のない)影を使用。
6	T	F	T	F			**Mesh, Contours** サーフェスをメッシュ、棒を輪郭で表示(等高線が分布(distribution)レベルの境界線を表示)。
7	T	F	F	T			**Mesh, Zones** サーフェスをメッシュ、棒を輪郭で表示。ゾーン色を線に使用。
8	F	T	T	F			**Shaded, Contours** フラットな(濃淡のない)影と等高線でサーフェスと棒を表示。
9	F	T	F	T			**Shaded, Zones*** ゾーン色がサーフェスと棒を塗りつぶして表示。

Graph Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface	Bar	Comments
10	**F**	**F**	**T**	**T**			**Contours, Zones** **等高線とゾーン色が分布(distribution)レベルを表示。**
11	T	T	F	T			**Mesh, Shaded, Zones** **9番のグラフのサーフェスにメッシュ、棒に輪郭を追加。**
12	T	T	T	F			**Mesh, Shaded, Contours** **5番のグラフに等高線を追加。**
13	F	T	T	T			**Shaded, Contours, Zones** **9番のグラフに等高線を追加。**
14	T	F	T	T			**Mesh, Contours, Zones** **7番のグラフに等高線を追加。**

Graph Type	DrawMesh	DrawShaded	DrawContours	DrawZones	Surface	Bar	Comments
15	T	T	T	T			**Mesh, Shaded, Contours, Zones** **全ての基本形の総合。**

* このリリースでは、棒グラフのDrawZonesは、DrawZonesとDrawShadedの組み合わせと同じになっています。DrawZonesの組み合わせについては、将来変更されることがありますので、この図のグラフを作成するときは、DrawZonesとDrawShadedの組み合わせを使用してください。

4.2 棒グラフとヒストグラム

XtNxrt3dTypeがXRT3D_TYPE_BARである時、XtNxrt3dSurfaceDataで指されているデータは棒グラフに表示されます。各データポイントは一本の棒としてグラフ化されます。規則的または不規則的なグリッドのどちらのデータも棒グラフに表示できます。どちらの場合も、グリッドの間隔がそのまま使用されます。

棒のZ原点

棒はZ原点から出発し、この原点はXtNxrt3dZOriginでコントロールできます。原点のデフォルト値は0.0です。DrawShadedがTrueの場合、原点より大きい値の棒はXtNxrt3dSurfaceTopColorで表示されます。負の棒(原点より小さい値)にはXtNxrt3dSurfaceBottomColorが使用されます。

棒のスペース

棒がグラフに占めるスペース(可能範囲に対するパーセントで指定)は、XtNxrt3d[XY]BarSpacingを使用してコントロールできます。デフォルトは80%です。この値を小さくすると棒の幅が細くなり、100%(最大限)にすると棒がお互いに隣接します。

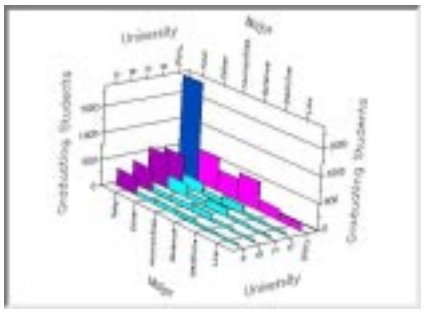


図21　X Spacingが1%、Y Spacingが100%の棒グラフ

ヒストグラム

ヒストグラムを作成するには、XtNxrt3d[XY]BarFormatをXRT3D_BAR_HISTOGRAMにセットします。(デフォルトはXRT3D_BAR_FIXEDです。) X軸、Y軸は、それぞれ_FIXEDか_HISTOGRAMのフォーマットに切り替えることができます。X軸がヒストグラムへ切り替えられた場合は、各棒の左端が対応するX値に揃うように描かれます。棒の幅は次のX値までの距離になります。各棒の幅は隣接する棒との間隔から計算されるので、データを_FIXEDで表示した場合より、棒の数が一本少なくなります。

ヒストグラムは通常、不規則なグリッド・データに使用されます。尚、不規則なグリッド・データの詳細は6.2をご覧ください。

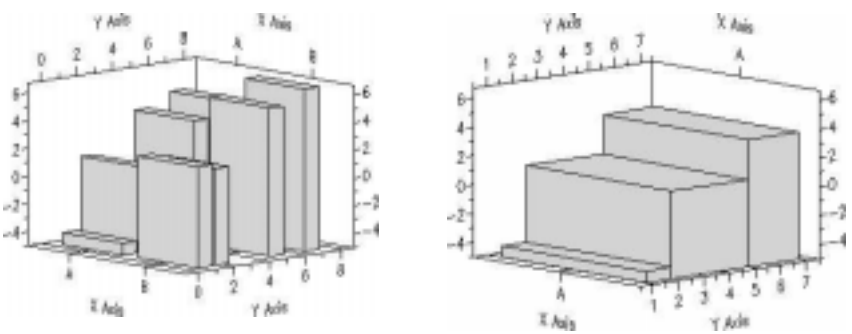


図22　同じデータを_FIXEDと_HISTOGRAMで表示した場合

棒の色

DrawShadedがTrueで、DrawZonesがFalseの場合は、各棒の色をコントロールすることができます。通常、原点より小さい値を持つ棒はsurface bottomの色、それ以外はsurface topの色が使用されます。その他、棒の行や列または各棒の色をそれぞれ指定することもできます。

XtNxrt3dXYColorsは、棒の色の指定に使用されます。このリソースは、次の構造体のポインタのNULL終端リストのポインタを取ります。

```
typedef struct {
  int             xindex;  /* 0-indexed  */
  int             yindex;  /* 0-indexed  */
  char *          color;   /* 色名 */
  unsigned long   res;    /* reserved  */
} Xrt3dXYColor
```

各棒の色は、構造体内のインデックスを指定することにより変更できます。例えば、三番目のXデータ行の二番目のポイントの棒を赤にセットする場合、Xrt3dXYColor構造体で値(2, 1, “red”)を指定します。棒の行全体の色を指定するには、インデックスを-1にセットしてください。(例: 五番目のX行の棒全体を緑に設定するには、(4, -1, “green”)とします。) 全ての棒の色を設定するには、*xindex*と*yindex*の両方を-1にセットしてください。

*Xrt3dGetXYColor()*を呼び出して、棒に使用されている色を取り出すことができます。また、*Xrt3dSetXYColor()*を使用すると棒の色の変更が簡単に行えます。例えば、次のようにして(Y方向の)四番目の行の棒を黄色に変更することができます。

```
Xrt3dSetXYColor(mygraph, -1, 3、"Yellow")
```

XRT/3dは、一つの組み合わせ(*xindex*、*yindex*)につき、一つのエントリーしか保持できません。同じインデックスに二番目のエントリーが与えられると、一番目のエントリーは削除されます。後から加えられたエントリーが元のエントリーより優先します。

Xrt3dXYColor構造体で*color*がNULLにセットされると、そのインデックスのエントリーを削除するリクエストとみなされます。これは、*Xrt3dSetXYColors()*で色のエントリーを個別に削除する時に使用できます。

XYColorsは凡例には現れません。この方法は、棒を特定の色でラベル付けすることが重要でない時に使用してください。ラベル付けが重要な時は、4Dの棒グラフを作成してください。4Dの棒グラフはセクション8.2に解説があります。

bars.c

簡単な棒グラフのサンプル・プログラムのソースコードが、*$XRTHOME/src/3d/examples*にあります。このプログラムでは、ユーザが各棒をクリックすることによりその色を変更することができます。図23は、*bars.c*がコンパイル、実行された結果です。

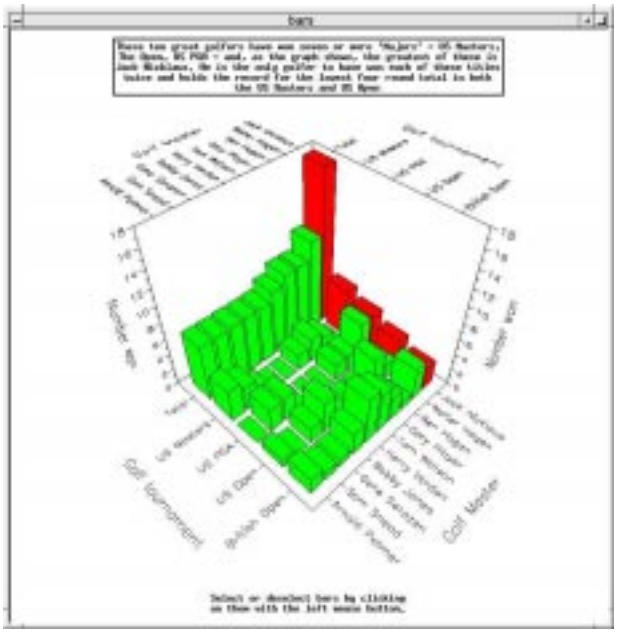


図23　bars.cで作成されたウィンドウ

4.3 等高線とゾーンの表示

DrawContoursまたはDrawZonesが`True`の時、XRT/3dは、組み込まれている100の等高線スタイルの配列を用いて各分布(distribution)レベルを区分けします。等高線をより良く表示するために、カラーまたはgreyscale(モノクロも含む)のXサーバでは、異なった等高線スタイルの配列が使用されます。

各等高線スタイルは、線のスタイル、幅、パターン、色、ゾーン色(各*level*を区分)に関する情報を持っています。等高線スタイルは、アプリケーションによってカスタマイズすることもできます。詳細は、8.6をご覧ください。

使用されるスタイル

XRT/3dは、図24にあるように、分布レベル全体に平均して等高線スタイルが使用されるよう、各レベルのスタイルを自動的に決定します。

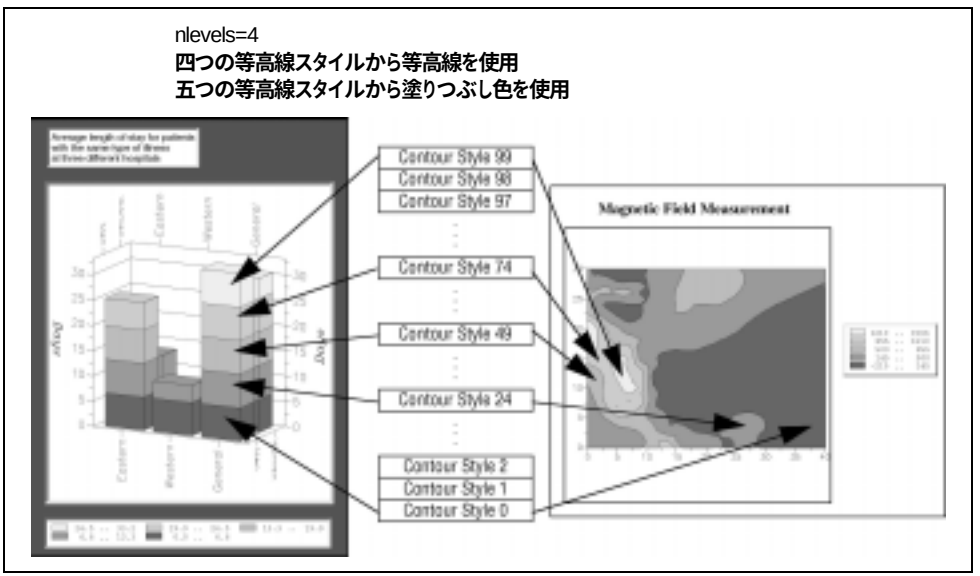


図24　サンプリングされている等高線スタイル

等高線とゾーンのプロジェクション(投影)

`XtNxrt3dProjectZMax`と`XtNxrt3dProjectZMin`を使用して、等高線とゾーンは、ユニット・キューブの上部または下部に表示することができます。これらのリソースは、次の定数から組み合わされた整数を取ります。

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

例えば、*z=zmax*の面に等高線とゾーンのプロジェクションを表示するには、`XtNxrt3dProjectZMax`を値
(`XRT3D_PROJECT_CONTOURS|XRT3D_PROJECT_ZONES`)に設定します。全てのプロジ

ェクションを削除するには、このリソースを0にセットします。

これらのリソースは、DrawContoursまたはDrawZonesの値に依存していません。但し、等高線の作成や表示に影響するリソース(XtNxrt3dContourStyles、XtNxrt3dNumDistnLevels、XtNxrt3dZoneData等)は、等高線とゾーンのプロジェクションに影響します。XtNxrt3dProjectZMaxとXtNxrt3dProjectZMinは、2Dグラフや棒グラフ、スキャッター・プロットでは無視されます。

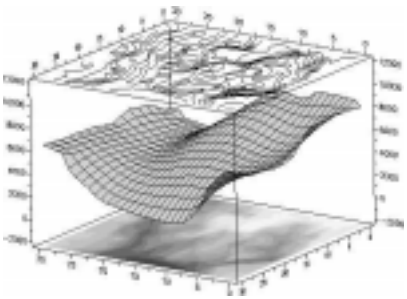


図25　等高線とゾーンのプロジェクション

ゾーンの方法

XtNxrt3dZoneMethodリソースを使用して、各ゾーンの表示をコントロールできます。デフォルト(XRT3D_ZONE_CONTOURS)では、XRT/3dは、各等高線のインターバル(間隔)を塗りつぶします。XRT3D_ZONE_CELLSにセットされると、グリッドで構成されるセルの四つの角の平均から色を決定し、各セルを塗りつぶします。図26で、その違いが確認できます。

_CELLSによるゾーンは粗いイメージのサーフェスを作成しますが、_CONTOURSの場合に比較してパフォーマンスが良くなります。但し、グリッドが大きくなれば、ビジュアル面での差異は小さくなります。

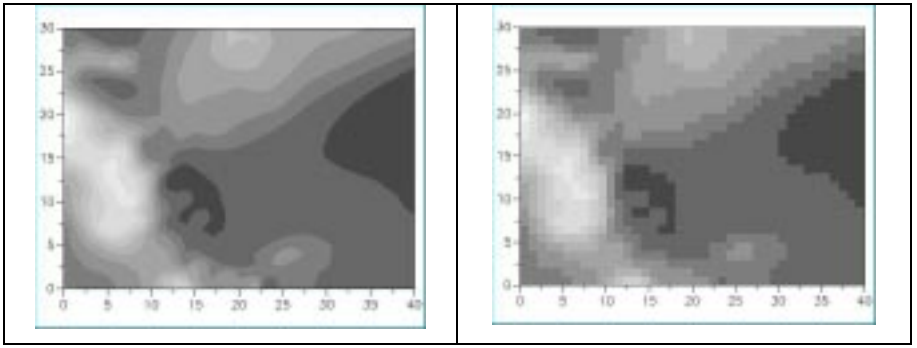


図26　等高線(_CONTOURS)とセル(_CELLS)によるゾーン

4.4　メッシュのコントロール

メッシュの色

DrawMeshがTrueの時のメッシュの下部と上部の色は、XtNxrt3dMeshBottomColorとXtNxrt3dMeshTopColorで設定できます。デフォルトの色は黒です。色の設定については2.7を参照してください。

メッシュのフィルタ

XtNxrt3d[XY]MeshFilterは、表示する前にメッシュをどのようにフィルタにかけるかを指定します。デフォルトではフィルタは実行されません。0にセットされている場合は、XRT/3dが自動的にメッシュを最適な状態にフィルタし、グラフのスケーリングやデータの変更に合わせて、フィルタの度合いを変更します。

これらのリソースを正の整数に設定して、メッシュのフィルタをハード・コーディングすることができます。例えば、値5は、5番目毎の線が描かれるようにメッシュをフィルタします。

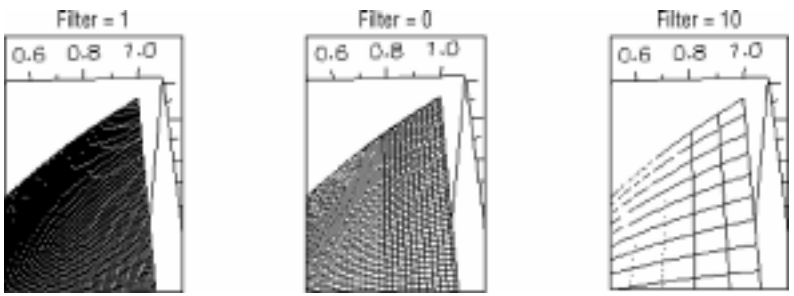


図27　メッシュのフィルタ

隠線の消去

DrawMeshがTrueでDrawShadedがFalseの時、他の部分で隠されたグリッドと等高線はデフォルトで表示されません。これらの線を表示するには、XtNxrt3dDrawHiddenLinesをTrueにしてください。

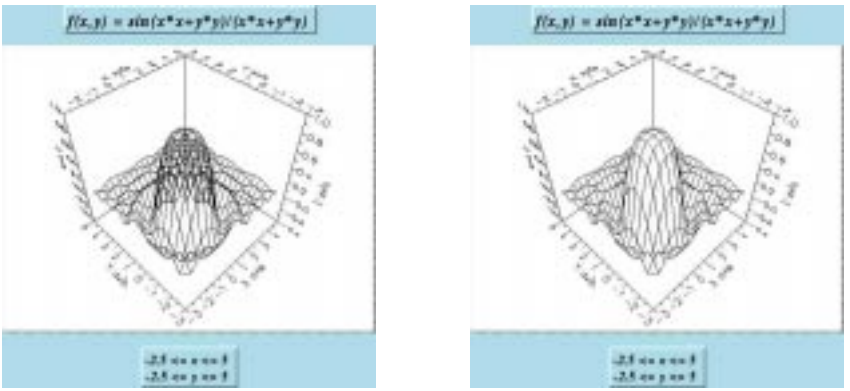


図28　隠線の消去

4.5　サーフェスの色

DrawShadedがTrueの時のサーフェスの下部と上部の色は、XtNxrt3dSurfaceBottomColorとXtNxrt3dSurfaceTopColorで設定できます。デフォルトでは、下部の色は"dim grey"、上部は"light grey"です。色の設定については2.7を参照してください。

4.6　サーフェスの塗りつぶし

XtNxrt3dSolidSurfaceをTrueにすると、XRT/3dはサーフェスの端とZ軸の最小値の面を結んで、側面を描きます。(図29参照)

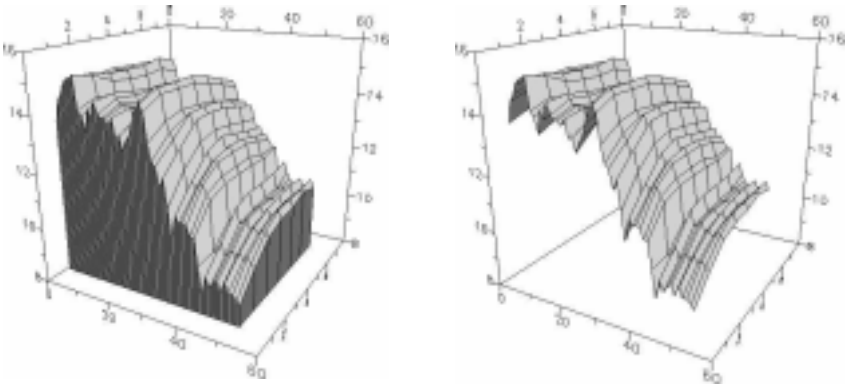


図29　Solid Surfaceをオンとオフにした場合

5

XRT/3dのプログラミング: スキャッター・プロット

スキャッター・プロットの三つの基本タイプ

シンボルとドロップ・ラインのスタイル ■ データ・スタイル

5.1 スキャッター・プロットの三つの基本タイプ

XtNxrt3dSurfaceDataを使用してデータがウィジェットに渡され、XtNxrt3dTypeの値がXRT3D_TYPE_SCATTERであると、スキャッター・プロットが表示されます。スキャッター・プロットは、一つまたは複数の(x, y, z)ポイントのシリーズ(series)を表示します。通常、スキャッター・プロット・データはXRT3D_DATA_POINTで指定します。この*ポイント・データ(point data)*は、任意の数のシリーズをサポートします。

各シリーズは、他のシリーズとは異なるシンボルを持つことができます。XRT3D_TYPE_SCATTERがグリッド・データ(XRT3D_DATA_GRIDまたはXRT3D_DATA_IRGRID)と一緒に使用されると、各グリッド・ポイントがスキャッター・プロットのポイントを定義します。この場合、一つのデータ・シリーズだけが使用されます。XRT/3dには、3Dスキャッター・プロット、ドロップ・ライン付きの3Dスキャッター・プロット、及び2Dスキャッター・プロットの三つの種類があります。

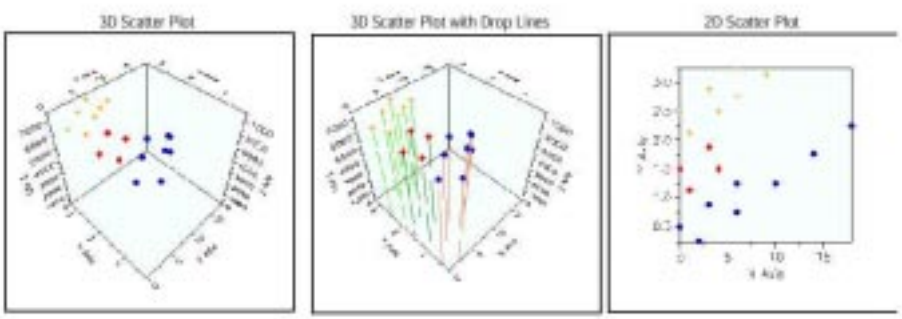


図30　スキャッター・プロットの例

ドロップ・ラインは、3Dスキャッター・プロット内の各ポイントからプロット・キューブのZ軸の原点の面上でマッチする(x, y)点に描画される線です。(x, y, z)ポイントと(x, y, origin)をつなぐドロップ・ラインを表示するには、XtNxrt3dDrawDropLinesをTrueにしてください。。このラインは2Dスキャッター・プロットでは描画されません。

2Dスキャッター・プロットは、x値とy値のみを使用し、z値は無視されます。このグラフは、XtNxrt3dDrawMeshとXtNxrt3dDrawShadedをFalseにして表示することができます。(サーフェスや棒グラフの場合と同じです。)

5.2 シンボルとドロップ・ラインのスタイル

スキャッター・プロットのシンボルとドロップ・ラインは、Xrt3dDataStyleで指定されます。各シリーズに対して必ず一つのデータ・スタイルが必要です。一番目のシリーズは一番目(インデックス0)のデータ・スタイル、二番目のシリーズは二番目のデータ・スタイルを使用します。シリーズの数より提供されているデータ・スタイルの数が少なければ、余ったシリーズにはデフォルトのデータ・スタイルが使用されます。

線のパターン、色、幅はXrt3dLineStyle、シンボルの色、サイズ、パターンはXrt3dSymbolStyleでコントロールされます。

5.3 データ・スタイル

データ値が表示された時にどのように見えるか(即ち、色、線パターン、シンボル・スタイル、線幅等)は、そのデータ値に定義されているデータ・スタイルに依存します。

デフォルトのデータ・スタイル

デフォルトのデータ・スタイルは、モノクロかカラー両方のサーバに対して用意されています。XtNxrt3dDataStylesUseDefaultがTrueであれば、グラフは、使用しているサーバに適したデフォルトのデータ・スタイルを使用します。データ・スタイルを明示的にセットした後でも、プログラムは、XtNxrt3dDataStylesUseDefaultをTrueにセットすることによりデフォルトのスタイルに戻ることができます。

Xrt3dDataStyle構造体は、*<Xm/xrt3dcom.h>*に定義されています。

```
typedef struct {
 Xrt3dLinePattern     pattern;/* line pattern */
 XrtColor             color; /* color */
 int                  width; /* line width */
} Xrt3dLineStyle;

typedef struct {
 Xrt3dSymbolPattern   pattern;/* symbol pattern */
 XrtColor             color; /* color */
 int                  size; /* symbol size */
} Xrt3dSymbolStyle;

typedef struct {
 Xrt3dLineStyle       line_style; /* line style */
 Xrt3dSymbolStyle     symbol_style;/* symbol style */
} Xrt3dDataStyle;
```

Xrt3dDataStyleは、データをどのように表示するかという情報を持った構造体です。このフィールドは下記の通りです。

Pattern	*<Xm/xrt3dcomh>*に定義されている、ドロップ・ラインの線パターンまたは、ポイントのシンボル・パターン。線パターンの場合は図31にあるXRT3D_LPAT定数、シンボル・パターンは図32にあるXRT3D_SYMBOL定数です。
Color	ドロップ・ラインまたはスキャッター・シンボルの色。色は色名か、RGBを表す#で始まる16進数で表示します。(2.7参照)
Width	ドロップ・ラインに使用される線幅。値は必ず1以上にしてください。
Size	Pixelで表すスキャッタ・シンボルの直径。値は必ず1以上にしてください。

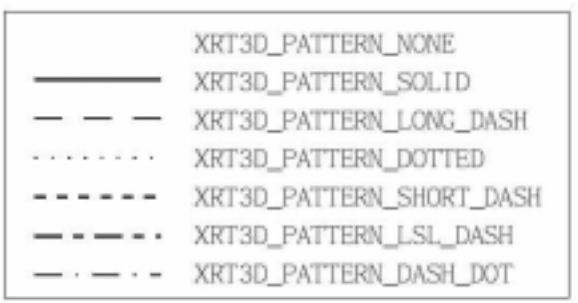


図31　線のパターン

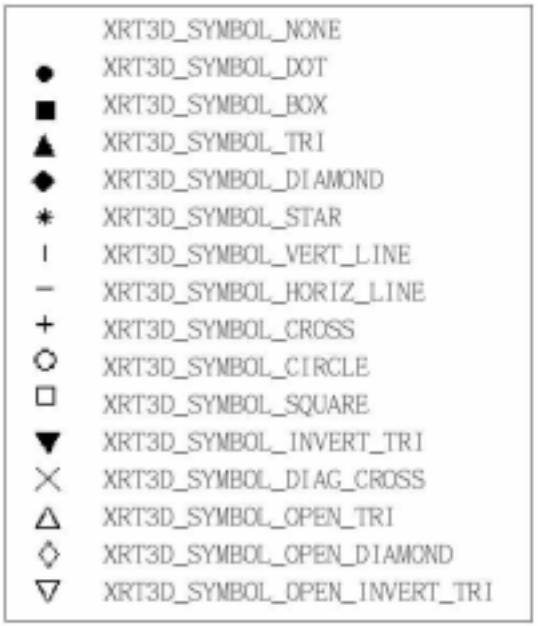


図32　シンボルのスタイル

データ･スタイルのセット

プログラムでデータ･スタイルを明示的にセットする場合、使用される環境を考慮してデータ･スタイルをセットしてください。例えば、モノクロの画面では、赤の棒と緑の棒は同じように表示されてしまいます。

データ･スタイルをセットすると副作用で、`XtNxrt3dDataStylesUseDefault`が`False`にセットされます。スタイルがセットされていないデータには、デフォルトのスタイルが適用されます。図33は、標準的なデータ･スタイル設定の例です。

```
Static Xrt3dDataStyle scatter_styles[] = {
/* 0 */ {{ XRT3D_LPAT_SOLID, "coral", 1 },
   { XRT3D_SYMBOL_DOT, "medium blue", 7 }},
/* 1 */ {{ XRT3D_LPAT_SOLID, "forest green", 1 },
   { XRT3D_SYMBOL_DIAMOND, "red", 7 }},
/* 2 */ {{ XRT3D_LPAT_SOLID, "lime green", 1 },
   { XRT3D_SYMBOL_STAR, "orange", 7 }},
/* 3 */ {{ XRT3D_LPAT_SOLID, "royal blue", 1 },
   { XRT3D_SYMBOL_TRI, "magenta", 7 }}
};

static Xrt3dDataStyle *my_dstyles_ptr[] = {
   &scatter_styles[0],
   &scatter_styles[1],
   &scatter_styles[2],
   &scatter_styles[3]
};
....
XtVaSetValues(graph, XtNxrt3dDataStyles, my_dstyles_ptr, NULL);
```

図33 データ･スタイルのセット

データ･スタイルへのアクセス

データ･スタイルはXtNxrt3dDataStylesリソースか、または*Xrt3dSetNthDataStyle()*、*Xrt3dGetNthDataStyle()*でアクセスすることができます。

XtNxrt3dDataStylesは、データ･スタイル配列全体を取得または設定することができます。*XtGetValues()*と一緒に使用された場合、返された配列ポインタはread-onlyです。*Xrt3dDupDataStyles()*は、返されたデータ･スタイルの配列ポインタをコピーする時に使用できます。例えば、全てのドロップ･ラインの幅を5pixelにする時は、次のようにしてください。

```
void SetThickLines(Widget graph)
{
 int i;
      Xrt3dDataStyle **ds, **myds;
      XtVaGetValues (graph, XtNxrt3dDataStyles, &ds, NULL);
      myds = Xrt3dDupDataStyles(ds);
      for(i=0; myds[i]; i++) {
          myds[i]->line_style.width = 5;
      }
      XtVaSetValues(graph, XtNxrt3dDataStyles, myds, NULL);
      /* Release this copy once styles are set */
      Xrt3dFreeDataStyles(myds);
}
```

プログラムは、*Xrt3dGetNthDataStyle()*を使用して特定のXrt3dDataStyle構造体のポインタを取得することができます。例えば、graphがスキャッター・データのグラフであれば、次のコードはデータ・シリーズに使用されているシンボルのサイズをプリント・アウトし、それを二倍にします。

```
void DoubleSymbolSize(Widget graph, int series)
{
 Xrt3dDataStyle *dstyle, my_dstyle;
 dstyle = Xrt3dGetNthDataStyle(graph, series);
 printf(斗ine width for series %d of points: %d\n・ series,
     dstyle->symbol_size.size);
/* Treat dstyle as a 途ead-only・ptr. Modify a local copy */
 my_dstyle = *dstyle;
 my_dstyle.symbol_style.size *=2;
 Xrt3dSetNthDataStyle(graph, series, &my_dstyle);
}
```

6

XRT/3dデータ

データについて

Xrt3dData構造体

6.1 データについて

グラフ化されるデータは、Unixファイル、データベース、リアルタイム・データのフィード、または他のマシンで実行されている処理等、あらゆるソースから持ってくることができます。XRT/3dは、実験や分析の結果等のデータをビジュアルに表示することができます。

この章では、`Xrt3dData`構造体を解説し、この構造体の割り当てやロードの例を幾つか紹介します。

ガイドライン

XRT/3dは増加するXとYのデータのサーフェスをグラフ化しますが、(球体等)XまたはY方向に折り返すデータは扱うことはできません。また、XRT/3dは、データが規則的または不規則的なグリッド・サーフェス形式に従っている場合にグラフを表示します。XとYのグリッド線がメッシュを構成している時、線が交差する場所で、Z値を持つポイントを定義します。ポイントにZ値がない場合は、そのポイントは*hole* (穴)と呼ばれます。

スキャッター・プロットでは、一つまたは複数の(x, y, z)ポイントのシリーズとして提供されたデータをグラフ化します。

XRT/3dは小さな次元のメッシュで最適化されます。大きい次元のデータは、グラフの作成に時間がかかり、あまり良い結果が得られません。大きい次元のデータを扱う場合は、グラフ化する前に、データを減らしてください。(例:グリッド・ポイントを10個目ごとに`Xrt3dData`構造体にロードする。または*Xrt3dDataWindow()*を使用する。)

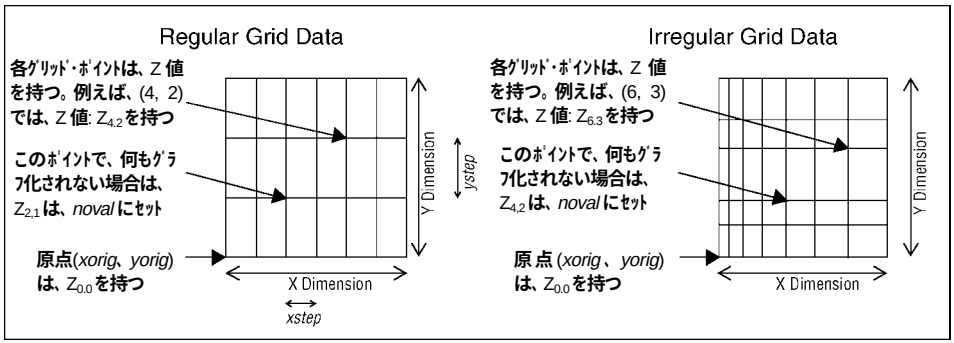


図34　規則的な7x4のグリッドと不規則的な8x6のグリッド

データのhole

Xrt3dData構造体は、*noval*(no value;値なし)と呼ばれるエレメントを持ちます。XRT/3dは、グリッド･データに*noval*と同じ値がある場合は、そのグリッド線の交差点を*hole*として扱います。*noval*は、データの範囲より充分離れた値にセットしてください。XRT3D_HUGE_VAL (*xrt3d_common.h*に定義されています。)にセットすることもできます。

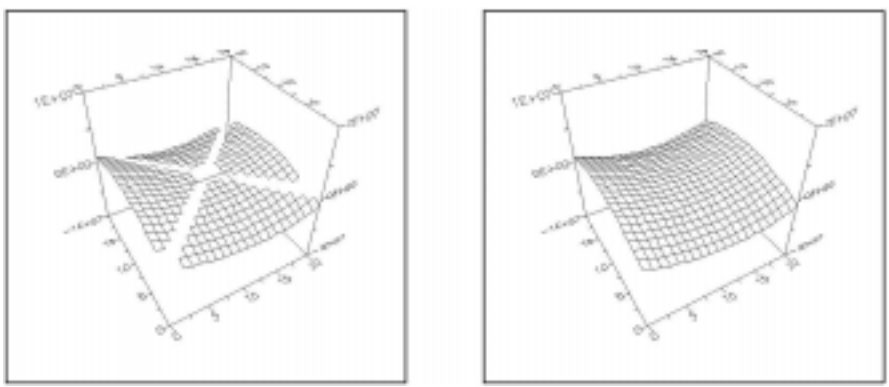


図35　holeの例(左のグラフの*noval*は0にセットされている)

ファイルのデータ

データがファイルにあり、リアルタイムに変更やアップデートする必要がない場合は、二つの方法があります。一つは、ファイルが*Xrt3dMakeDataFromFile()*が理解できる構文に従うようデータを操作する方法です。この手続きは、Xrt3dData構造体を割り当て、指定されたファイルのデータをロードします。*Xrt3dMakeDataFromFile()*の解説はAppendix Cにあります。*Xrt3dSaveDataToFile()*という手続きもあります。

もう一つは、*Xrt3dMakeGridData()*または*Xrt3dMakeIrGridData()*を用いてXrt3dData構造体を割り当て、(*fgets(3)*または*fscanf(3)*を使用して)データ･ファイルから読み込まれたデータでセットする方法です。

データの変更

データは、他のウィジェットのリソースを変えずに、変更することができます。Xrt3dData構造体は幾つでも作成することができ、XtNxrt3dSurfaceDataリソースが現在のデータを指すように設定して、構造体を切り替えることができます。

リアルタイム・データ

データがリアルタイムに変化し、それに従ってアップデートされるグラフの場合は注意が必要です。Xt Intrinsicsには、リアルタイム・イベントに対応するよう、アプリケーションのコールバック・ルーチンを呼び出す機能が提供されています。

*XtAppAddInput()*は、ファイル記述子にペンディングされている入力がある時に呼び出される、アプリケーションのコールバック・ルーチンの登録に使用されます。ファイル記述子は、ファイル、パイプ、または他のマシンで実行されている処理のソケット・コネクションとすることができます。

他には、アプリケーションが*XtAppAddTimeOut()*を使用して、インターバル・タイマーを定義する方法があります。指定されたインターバルが切れる度に、アプリケーションのコールバック・ルーチンが実行されます。この時、アプリケーションはグラフ化するデータが他にないかを調べ、その結果に従って処理を行うことができます。

これらのルーチンの詳しい情報については、Xt Intrinsicsの資料をご覧ください。

リアルタイムのパフォーマンス

リアルタイムの最良のパフォーマンスを得るためには、グリッドの次元を小さく保つようにしてください。DrawShadedとDrawZonesに比べて、DrawMeshではグラフの表示が速く行われます。XtNxrt3dDrawHiddenLinesがTrueの場合は、XRT/3dは最適化されたアルゴリズムを使用し、スピードが速くなります。また、ウィンドウのサイズを小さく保ち、ダブル・バッファリングをオフにしたり、XtNxrt3dRepaintを使用すると、パフォーマンスが向上します。

データの役割

Xrt3dData構造体を割り当てたり、破棄するのは、アプリケーションの役割です。XRT/3dはデータのコピーを作成しませんが、XtNxrt3dSurfaceDataとXtNxrt3dZoneDataに指されているアプリケーションのメモリ内のデータを参照します。

XRT/3dがウィンドウを再描画する場合は、XtNxrt3dSurfaceDataで指されているデータを使用します。アプリケーションでは、Xrt3dData構造体が使用されている間でも、構造体内の値を変更することができます。但し、次にXRT/3dが要求した時に、必ず正しい構造体が渡されるようにしてください。データ値を変更した後は、XRT/3dが表示をアップデートするように、XtNxrt3dSurfaceDataのリソースをリセットする必要があります。

6.2 Xrt3dData構造体

XRT/3dがグラフ・イメージを表示する時は、XtNxrt3dSurfaceDataに指されているXrt3dData構造体のデータ値を使用します。

Xrt3dData構造体は、*<Xm/xrt3d_common.h>*に定義されています。(Appendix Eにも解説されています。)Xrt3dDataは、regularとirregularなグリッド構造体のunionとして定義されています。

```
typedef union{
 Xrt3dGridData   g; /* 規則的なグリッド */
 Xrt3dIrGridData ig; /* 不規則的なグリッド */
 Xrt3dPointData   p; /* スキャッター・プロット*/
} Xrt3dData;
```

6.2.1 規則的なグリッドのデータ

Xrt3dGridData構造体は次のように定義されています。

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

Xrt3dGridData は 、 *Xrt3dMakeGridData()* と *Xrt3dMakeDataFromFile()* によって XRT3D_DATA_GRIDにイニシャライズされるtypeメンバを持っています。

*numx*と*numy*はXとYのグリッド線の数を定義する整数です。サーフェスとヒストグラムでは、*numx*と*numy*は≥2でなければなりません。棒グラフの場合の*numx*と*numy*は≥1です。

XRT/3dが"no value"と扱う値は*noval*で指定されます。XとYのグリッド線の間隔は*xstep*と*ystep*で指定され、これらは常に>0です。グリッド全体の原点は、*xorig*と*yorig*で指定されます。

*values*は、図36にあるように、各グリッド・ポイントのZ値の2次元配列です。

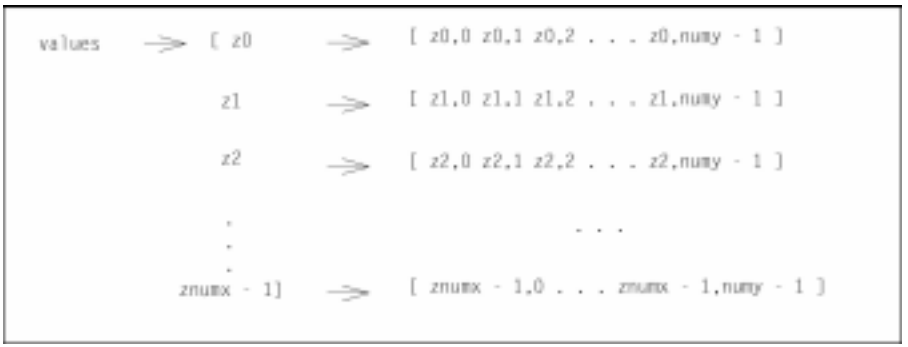


図36　Z値の2次元配列

Xrt3dData構造体は、*Xrt3dMakeGridData()*を使用して簡単に割り当てることができます。この手続きはAppendix Cに解説されています。

6.2.2　不規則的なグリッドのデータ

Xrt3dIrGridData構造体は以下のように定義されています。

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

Xrt3dIrGridData構造体は、*Xrt3dMakeIrGridData()*と*Xrt3dMakeDataFromFile()*によってXRT3D_DATA_IRGRIDにイニシャライズされるtypeメンバを持っています。

*numx*と*numy*は、XとYのグリッド線の数を定義する整数です。サーフェスとヒストグラムでは、*numx*と*numy*は≥2でなければなりません。棒グラフの場合の*numx*と*numy*は≥1です。

XRT/3dが"no value"と扱う値は*noval*で指定されます。*xgrid*と*ygrid*は、XとYのグリッド線の値である*numx*と*numy*の長さを持つdoubleの配列です。

*values*は、各グリッド・ポイントのZ値の2次元配列です。

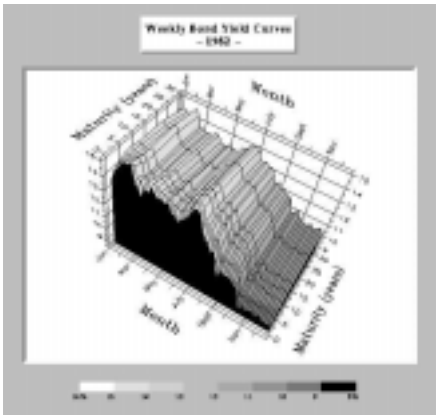


図37　Irregular サーフェス・プロット

値設定の例(グリッド・データ)

アプリケーションでは、簡単にXrt3dData構造体にアクセスして値を読み込んだり変更したりすることができます。次の例は、規則的なグリッドのサーフェスを42.0にイニシャライズしています。

```
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;
  }
}
```

6.2.3　スキャッター・プロットのデータ

Xrt3dPointData構造体は、次のように定義されます。

```
typedef struct {
      Xrt3dDataType    type; /* XRT3D_DATA_POINT */
      int      nseries;
      double       noval;
      Xrt3dPointSeries *series;
} Xrt3dPointData;
```

Xrt3dPointData構造体は、*Xrt3dMakePointData()*と*Xrt3dMakeDataFromFile()*によってXRT3D_DATA_POINTにイニシャライズされるtypeフィールドを持っています。

*nseries*はポイント・データ構造体内のシリーズの数を指定(常に≥1)します。hole値(no value)として処理される特別な値は*noval*で指定されます。*series*は、nseriesのXrt3dPointSeries構造体の配列(各シリーズに対して一つ)です。

Xrt3dPointSeries構造体は次のように定義されます。

```
typedef struct {
      int      npoints;
      Xrt3dPoint3D *points;
} Xrt3dPointSeries;
```

*npoints*はseries内のポイントの数、*points*はnpointsのXrt3dPoint3D構造体の配列(各ポイントに対して一つ)です。Xrt3dPoint3D構造体は次のように定義されます。

```
typedef struct {
      double       x, y, z;
} Xrt3dPoint3D;
```

下記は、Xrt3dPointData構造体を割り当てている例です。この例は、*Xrt3dMakePointData()*と*Xrt3dMakePointDataSeries()*という簡易手続きを使用しています。

```
/* Create all of the series */
Xrt3dPointData* pPointData = (Xrt3dPointData *)
Xrt3dMakePointData(nNumSeries, 1, XRT3D_HUGE_VAL, FALSE);

/* Allocate the points in a specific series */
Xrt3dMakePointDataSeries(&(pPointData->series[nSeries])
      , nNumPoints);
```

6.2.4 簡易手続き

XRT/3dには、Xrt3dData構造体の操作に使用できる幾つかの簡易手続きがあります。これらの手続きは、Appendix Cに解説されており、ソースコードは、*$XRTHOME/src/3d/util*にあります。

Xrt3dDataCopy()	Xrt3dData**構造体のコピーを作成する。**
Xrt3dDataShaded()	**既存の構造体から新しい**Xrt3dData**構造体を作成し、元のデータの**shaded relief map**(起伏図)でセットする。(このルーチンの例は、***shader***デモを参照) ポイント･データには適用されない。**
Xrt3dDataSmooth()	**各ポイントとそのポイントを中心にする周囲のポイントを平均化し、**Xrt3dData**構造体のデータをスムーズにする。ポイント･データには適用されない。**
Xrt3dDataUpdateDataValue()	Xrt3dData**構造体のデータ値を新しい値に変更する。**
Xrt3dDataWindow()	**元のデータからサブセットを抽出、キュービックまたはリニア･スプラインを用いてリサンプルし、既存の構造体から新しい**Xrt3dData**構造体を作成する。このルーチンの例は、***spline***デモを参照) ポイント･データには適用されない。**

7

ユーザ･インタラクションのプログラミング

デフォルトのユーザ･インタラクション ■ トランスレーションとアクションの概要
ユーザ･インタラクションの開始 ■ ユーザ･インタラクションのアップデート
ユーザ･インタラクションの終了 ■ グラフ･データとのインタラクション
ウィンドウのリサイズ

ここでは、XRT/3dのユーザ･インタラクション機能(どのようにユーザがグラフとインタラクションを行うことができるか、またアプリケーションはそれをどのようにコントロールするか)を解説します。

7.1 デフォルトのユーザ･インタラクション

デフォルトで、ユーザは、ユニット･キューブの回転、移動、スケーリング、ズームを行うことができます。(図38参照)

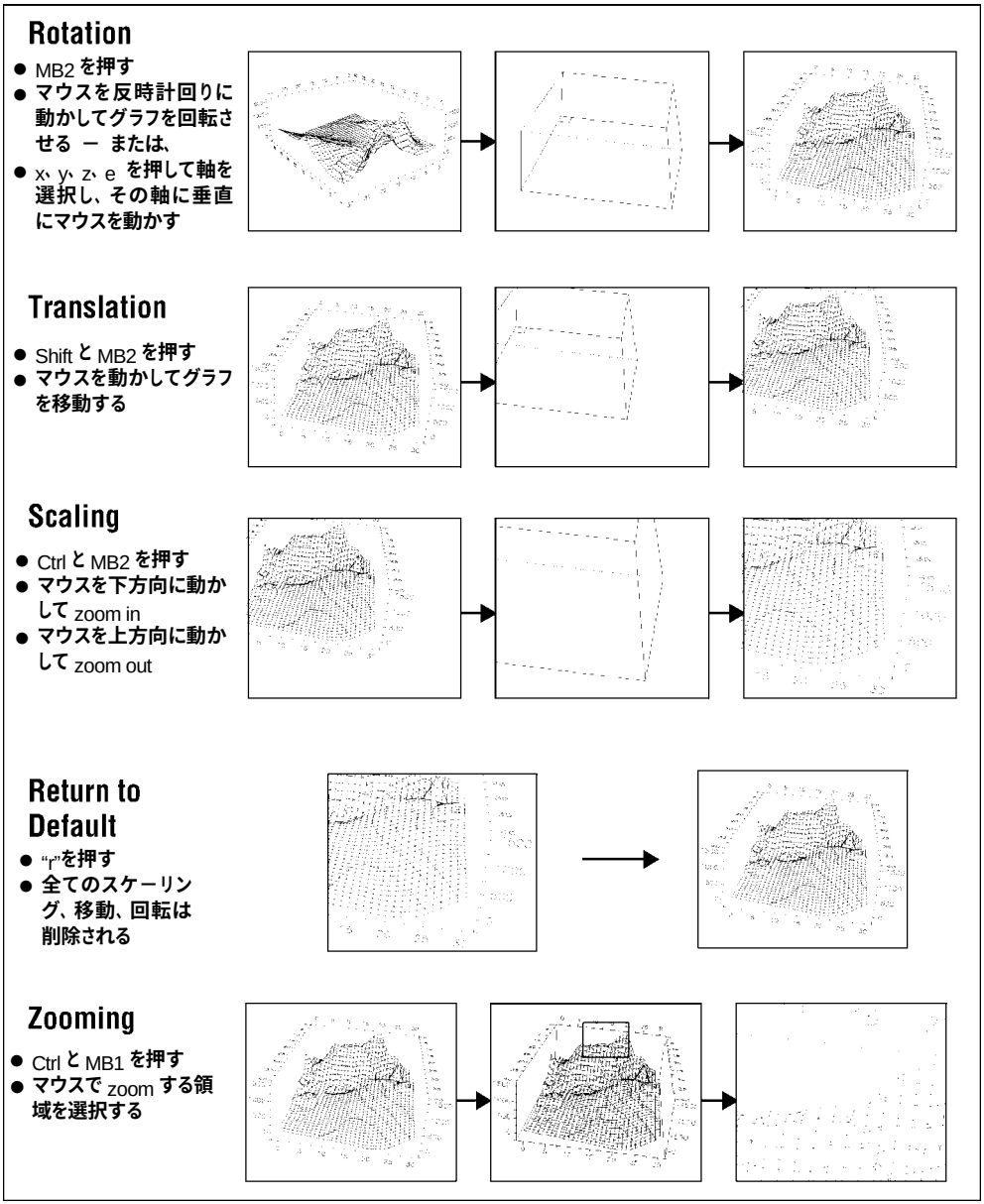


図38　XRT/3dのデフォルトのユーザ・インタラクション

7.2 トランスレーションとアクションの概要

XRT/3dのデフォルトのトランスレーション・テーブル(定義表)は、インタラクティブなアクションを引き起こすユーザ・イベントを定義しています。このユーザ・インタラクションは、次のようにしてカスタマイズすることができます。

- **トランスレーション・テーブル**　アプリケーションは、デフォルトのトランスレーションを変更したり、新しいアクションのトランスレーションを追加したりすることができます。Appendix Dには、XRT/3dのデフォルトのトランスレーションとアクションのリストがあります。
- **コールバック**　インタラクションの前、途中、後に呼び出されるコールバック手続きを登録することにより、アプリケーションは、ユーザがウィジェットとインタラクションを行った時に通知を受けることができます。コールバックは、各インタラクションを不許可にしたり、制限したりすることができます。

インタラクションの三つの段階

XRT/3dのデフォルトのユーザ・インタラクションには、次の三つの段階があります。

- ユーザ・インタラクションの開始
- ユーザ・インタラクションのアップデート
- ユーザ・インタラクションの終了

一つのインタラクションは、必ずこの段階を順番に通り、アプリケーションは各段階でコールバックによる通知を受けることができます。ここから次のセクションでは、各段階の詳細を解説していきます。

7.3 ユーザ・インタラクションの開始

*ModifyStart()*が呼び出され、ユーザ・インタラクションが開始されることをアプリケーションに通知します。図39は、このアクション・ルーチンの図解で、追加されたModifyコールバックがどこで呼び出されるかを示しています。

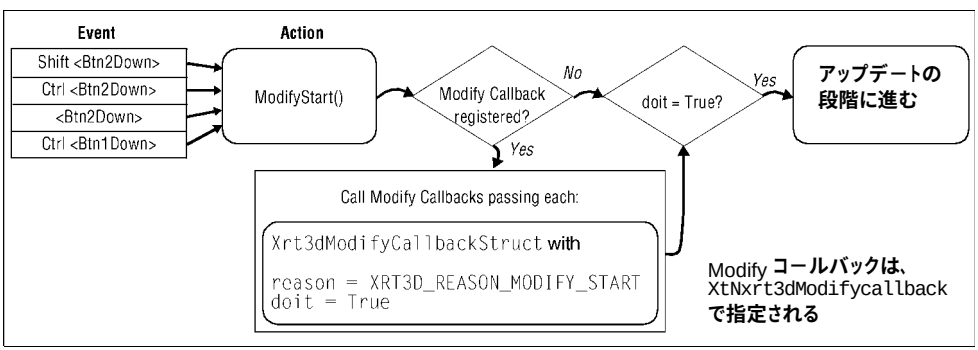


図39　*ModifyStart()*アクション・ルーチン

全てのユーザ･インタラクションを不可にする

インストールされているトランスレーション・テーブルに関らず、XtNxrt3dModifyCallbackを使用して全てのユーザ･インタラクションを不可にすることができます。各コールバック手続きには、次の構造体のポインタが渡されます。

```
typedef struct {
  int         reason;      // Read-only
  XEvent  *event;      // Read-only
  String  *params;     // Read-only
  Cardinal    nparams;     // Read-only
  Boolean doit;
} Xrt3dModifyCallbackStruct;
```

全てのユーザ･インタラクションを不可にするには、(*reason*パラメータがXRT3D_REASON_MODIFY_STARTの時に)この構造体の*doit*パラメータをFalseにしてください。*doit*がFalseの時、*ModifyStart()*は、次の*ModifyStart()*が呼び出されるまで、全てのアップデートのアクションを不可にします。

7.4 ユーザ･インタラクションのアップデート

幾つかのアクション･ルーチンの内の一つが呼び出され、ユーザ･インタラクションがアップデートされることをアプリケーションに通知します。インタラションが*ModifyStart()*アクションを通っていない場合、グラフはアップデートされません。

DrawPreview()またはDrawGraph()

ユーザがグラフを回転、スケーリング、移動する際、*DrawPreview()*または*DrawGraph()*のアクション･ルーチンを利用して、ビジュアルなフィードバックをユーザに提供することができます。*DrawPreview()*はデフォルトで呼び出され、ユニット･キューブの簡単な輪郭を描きます。アプリケーションで*DrawGraph()*を呼び出すように再定義し、ユニット･キューブ全体を描くように設定することもできます。但し､*DrawGraph()*は、小さなデータ･セットの場合に使用してください。

7.4.1 スケーリング、移動、ズーム

*ScaleUpdate()*はグラフのインタラクティブなスケーリング、*TranslateUpdate()*はインタラクティブな移動をアップデートし、*ZoomEnd()*は、(*ZoomUpdate()*で定義される)"zoom rubberband" で指定された箇所をズームします。これらのルーチンは、XtNxrt3dViewScaleとXtNxrt3dView[XY]Translateリソースを変更します。図40は、これらのアクション･ルーチンの図解で、追加したTransformコールバックがどこで呼ばれるかを示します。

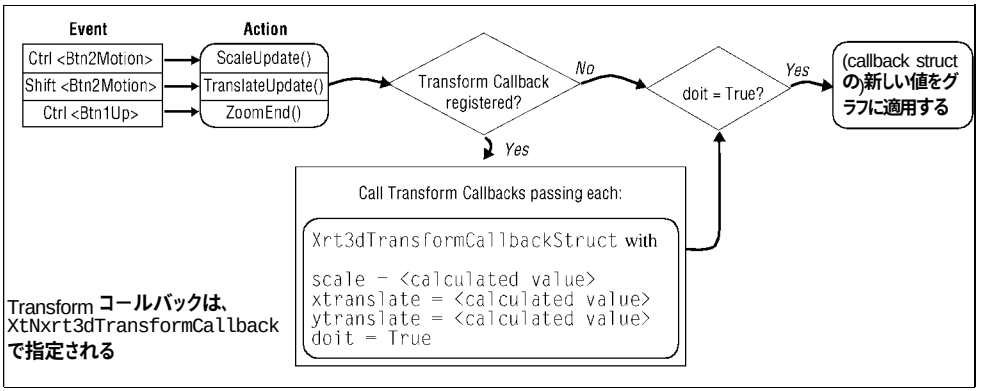


図40　*ScaleUpdate()*、*TranslateUpdate()*、 *ZoomEnd()*アクション・ルーチン

インタラクションのコントロール

スケーリング、移動、ズームは、XtNxrt3dTransformCallbackでコントロールできます。各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int         reason;      // Read-only
  XEvent  *event;      // Read-only
  String  *params;     // Read-only
  Cardinal    nparams;     // Read-only
  double  scale;
  double  xtranslate;
  double  ytranslate;
  Boolean doit;
} Xrt3dTransformCallbackStruct;
```

Transformコールバックは、*scale*、*xtranslate*、*ytranslate*、*doit*パラメータを変更することができます。*ScaleUpdate()*、*TranslateUpdate()*、*ZoomEnd()*は、これらのパラメータをグラフに適用します。図41では、Transformコールバックの*scale*パラメータを調べ、変更し、スケーリングを制限しています。

```
void
limit(Widget mygraph, XtPointer cd, XtPointer cbs)
{
    Xrt3dTransformCallbackStruct *tcbs = (Xrt3dTransformCallbackStruct *)cbs;

    if (cbs->scale < 0.5)  cbs->scale = 0.5;
    if (cbs->scale > 2.0)  cbs->scale = 2.0;
}
```

図41 スケーリングを0.5から2.0の間に制限するコールバック

上のコールバック・ルーチンを登録するには、次のコールを使用してください。

```
XtAddCallback(mygraph,
  XtNxrt3dTransformCallback, limit, NULL);
```

インタラクションのリセット

*ViewReset()*アクションは、XtNxrt3dViewScaleとXtNxrt3dView[XY]Translateリソースをデフォルト値に戻し、全てのスケーリングとトランスレーションを削除します。

*ViewReset()*は、登録されているTransformコールバックを開始し、*doit*がTrueにセットされたXrt3dTransformCallbackStructのポインタを渡します。アプリケーションは、コールバックで*doit*をFalseに設定して、リセットを拒否することができます。

7.4.2 回転

*RotateUpdate()*アクションは、グラフのインタラクティブな回転をアップデートします。このルーチンは、XtNxrt3d[XYZ]Rotationリソースを変更します。図42は、このアクション・ルーチンの図解で、追加されたRotateコールバックがどこで呼ばれるかを示します。

XtNxrt3dViewNormalized

XRT/3dは、デフォルトで、（軸のタイトルや注釈も含む)グラフ全体が表示されるようにユニット・キューブをリサイズします。この場合、ビューが回転されると、表示に支障をきたすことがあります。回転をスムーズにするには、回転が始まる前にXtNxrt3dViewNormalizedをFalseにセットしてください。

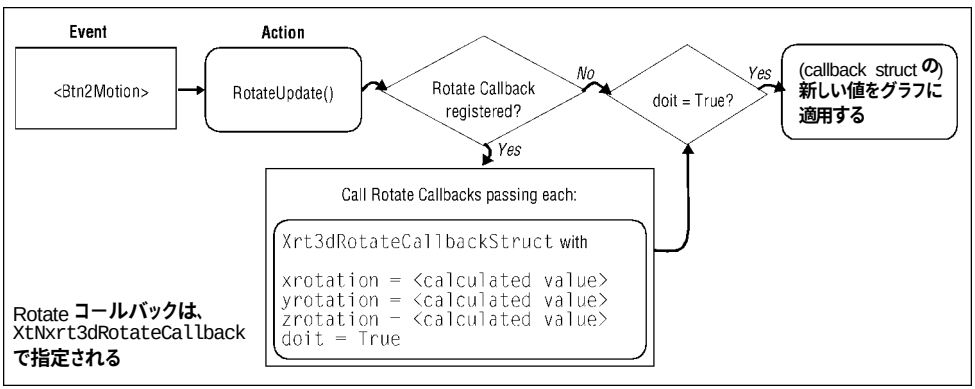


図42　*RotateUpdate()*アクション・ルーチン

回転のコントロール

XtNxrt3dRotateCallbackを使用して、回転をコントロールすることができます。各コールバックには次の構造体のポインタが渡されます。

```
typedef struct {
  int             reason;      // Read-only
  XEvent      *event;      // Read-only
  String      *params;     // Read-only
  Cardinal        nparams;     // Read-only
  double      xrotation;
  double      yrotation;
  double      zrotation;
  Boolean     doit;
} Xrt3dRotateCallbackStruct;
```

Rotateコールバックは、*xrotation*、*yrotation*、*zrotation*、*doit*パラメータを変更することができます。*RotateUpdate()*は、これらのパラメータをグラフに適用します。例えば、次のコールバックは、Y軸の回転を不許可にしています。(この例は、*view_dir*デモに使用されています。)

```
void
spin_and_tilt(
  Widget      widget,
  XtPointer   data,
  XtPointer   cbs
{
  Xrt3dRotateCallbackStruct *rotate_cbs;
  double  ay;

  rotate_cbs = (Xrt3dRotateCallbackStruct *) cbs;

  XtVaSetValues(widget,
     XtNxrt3dYRotation, &ay,
     NULL);
  rotate_cbs->yrotation = ay;
}
```

7.4.3 インタラクションの不許可

特定なユーザ･インタラクションを不可にするには、そのトランスレーションを*NoAction()*にセットする方法が一番簡単です。(7.3には、全てのインタラクションを不可にする方法が解説してありますので参照してください。) 次のコードは、回転とトランスレーションを不可にしています。

```
*.XtXrt3d.translations: #override \n\
  ~Shift ~Ctrl <Btn2Motion>:        NoAction() \n\
  Shift           <Btn2Motion>:       NoAction() \n\
```

または、コールバックを使用して、インタラクションを不許可にすることもできます。コールバックで、該当するインタラクションの渡されたコールバック構造体の*doit*パラメータをFalseにセットしてください。コールバックを呼び出したアクション･ルーチンは、このパラメータを調べ、*doit*がFalseであれば変更をグラフに適用しません。(図39と図42参照)但し、*ModifyEnd()*によって呼び出されたModifyコールバックは、アプリケーションにユーザ･インタラクションが終了したことを通知するだけですので例外です。

次のコールバックは、Xrt3dTransformCallbackStructの*doit*パラメータをFalseにセットして、ユーザがグラフをスケーリングできないようにしています。

```
void
deny_small_scale(Widget mygraph, XtPointer cd,
   XtPointer cbs)
{
  XrtTransformCallbackStruct *tcbs =
    (XrtTransformCallbackStruct *)cbs;

  if (cbs->scale < 0.1){
    cbs->doit = False;
  }
}
```

上のコールバック・ルーチンを登録するには、次のコールを使用してください。

```
XtAddCallback(mygraph,
  XtNxrt3dTransformCallback, deny_small_scale, NULL);
```

7.5 ユーザ・インタラクションの終了

*ModifyEnd()*アクションが呼ばれ、ウィジェットを再描画し(Exposeコールバックを始動)、ユーザ・インタラクションが終了したことをアプリケーションに通知します。アプリケーションは、`XtNxrt3dModifyCallback`を使用してこの通知を受けることができます。各コールバック手続きには、*reason*パラメータが`XRT3D_REASON_MODIFY_END`にセットされた`Xrt3dModifyCallbackStruct`構造体のポインタが渡されます。この構造体内のパラメータを変更しても、何も変化はありません。

7.6 グラフ・データとのインタラクション

グラフ・データとのユーザ・インタラクションを処理するには、アクション・ルーチン/コールバック(7.6.1参照)とイベント・ハンドリング手続き(7.6.2参照)の二つの方法があります。通常、アクション・ルーチンを使用する方が簡単です。

7.6.1 アクション・ルーチンとコールバック

トランスレーション・テーブルを使用して、ユーザがグラフ・データとインタラクションを開始しようとした時に、アプリケーションに通知することができます。アプリケーションは、クリックされたグラフ座標またはユーザ・イベントに一番近いデータ値を受け取ることができます。XRT/3dには、*Pick()*と*Map()*という二つのアクション・ルーチンが提供されています。これらのルーチンを使用するには、次のようにしてください。

- ユーザ・イベントをアクション・ルーチンへマップするトランスレーションを設定する。(どちらのルーチンにもデフォルトのトランスレーションがありません。)
- PickまたはMapコールバックを登録する。これらは、*Map()*または*Pick()*アクションの最中に開始され、*Xrt3dPick()*と*Xrt3dMap()*の結果が渡される。

XRT/3dのアクション・ルーチンの詳細は、Appendix Dに解説されています。

Pick()

*Pick()*は、ユーザ・イベントの位置に一番近いデータ値を決定し、それをコールバック手続きに渡します。*Pick()*は、*Xrt3dPick()*を使用します。この*Pick()*を用いて、ユーザがクリックした箇所に一番近いセットとポイントを見つけることができます。*Xrt3dPick()*の詳細はP80にあります。図43はこのアクション・ルーチンの図解で、どこでPickコールバックが呼び出されるかを示しています。

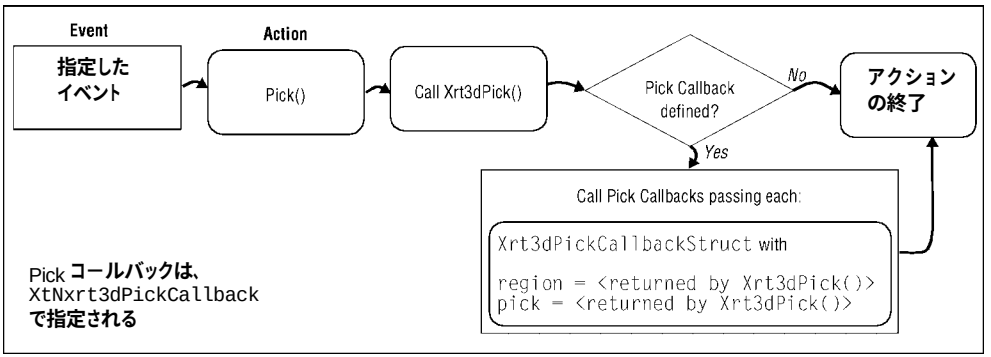


図43　Pick()アクション･ルーチン

*Pick()*は、それが呼び出されるためにイベントに割り当てられる必要があります。(次の例参照)

```
*.XtXrt3d.translations: #override \n\
     ~Shift ~Ctrl <Btn1Down>: Pick()
```

*Xrt3dPick()*の結果を受け取るには、XtNxrt3dPickCallbackを用いてコールバックを登録してください。各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int                    reason;      /* Read-only */
  XEvent               *event;      /* Read-only */
  String               *params;     /* Read-only */
  Cardinal                nparams;     /* Read-only */
  Xrt3dRegion     region;      /* Read-only */
  Xrt3dPickResult pick;        /* Read-only */
} Xrt3dPickCallbackStruct;
```

この構造体のフィールドを変更しても、何も変化しません。

Map()

*Map()*は、ユーザ･イベントのpixel座標にマップされたグラフ座標を決定し、それをコールバック手続きに渡します。*Map()*は、*Xrt3dMap()*を使用します。この*Map()*を使用して、どの値(x, y)をユーザがクリックしたかを見つけることができます。*Xrt3dMap()*はP82に解説されています。尚、*Map()*の働きは、図43に図解されている*Pick()*と同様です。

*Map()*は、それが呼び出されるために、イベントに割り当てられる必要があります。(次の例参照)

```
*.XtXrt3d.translations: #override \n\
  ~Shift ~Ctrl <Btn1Down>: Map()
```

*Xrt3dMap()*の結果を受け取るには、`XtNxrt3dMapCallback`を用いてコールバックを登録してください。各コールバックには、次の構造体のポインタが渡されます。

```
typedef struct {
  int                    reason;      /* Read-only */
  XEvent               *event;      /* Read-only */
  String               *params;     /* Read-only */
  Cardinal                nparams;     /* Read-only */
  Xrt3dRegion     region;      /* Read-only */
  Xrt3dMapResult  map;             /* Read-only */
} Xrt3dMapCallbackStruct;
```

この構造体のフィールドを変更しても、何も変化しません。

7.6.2 イベント・ハンドリングの手続き

Xt Intrinsicsは、マウスの動きやクリック、キーボードの入力等のイベント・ハンドリングを登録するメカニズムを提供しています。XRT/3dのプログラムは、このメカニズムを利用して、グラフ・ウィジェットに起こるイベントに対応することができます。

次のコードは、Xt Intrinsicsで、*my_event_proc()*を登録しています。グラフ・ウィンドウ上でマウス・ボタンのイベントが起こる度に、*my_event_proc()*が呼び出されます。

```
Widget my_graph;
XtAddEventHandler( my_graph,
  ButtonPressMask | ButtonReleaseMask,
  FALSE, my_event_proc, NULL);
```

イベント手続きは次のフォーマットを持ちます。

```
void
my_event_proc(
  Widget  w,
  caddr_t client_data,
  XEvent  *event
  )
{
  /* Deal with the event */
}
```

Xイベントのリクエストとハンドリングについては、Xt IntrinsicsやMotif関連の資料を参照してください。

XEvent構造体には、イベントのpixel座標があります。XRT/3dは、イベントのpixel座標を次の個所へマップする手続きを提供しています。

- イベント座標に一番近く表示されているグリッド・ポイントのインデックス
- イベント座標に位置するグラフ座標

Xrt3dPick()

*Xrt3dPick()*は、XRT/3dウィジェットのハンドル、(x, y)pixel座標、`Xrt3dPickResult`構造体のポインタを引数として取ります。

Xrt3dPickResult構造体は、(指定されたXRT/3dウィジェットの)指定されたpixel座標に一番近いグリッド・ポイントに関する情報でセットされます。

```
Xrt3dRegion
Xrt3dPick(Widget graph, Position pix_x, Position pix_y,
      Xrt3dPickResult pick)
```

Xrt3dPickResultは、次のように定義されます。

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

フィールドは次の通りです。

pix_x	*Xrt3dPick()*に渡されたx pixel座標。
pix_y	*Xrt3dPick()*に渡されたy pixel座標。
xindex	pixel座標に一番近いXグリッド線のインデックス。
yindex	pixel座標に一番近いYグリッド線のインデックス。
distance	渡されたpixel座標とデータ値 $Z_{(xindex, yindex)}$ の画面上の(pixel)距離。

*Xrt3dPick()*は、次の値の内一つを返します。

XRT3D_RGN_NOWHERE	与えられたpixel座標は、ピックできるどの箇所にも近くない。*xindex*と*yindex*は、-1にセットされる。
XRT3D_RGN_IN_GRAPH	与えられたpixel座標は、グラフ領域にある。*xindex*、*yindex*、*distance*は、正しくセットされる。データのholeがピックされた場合は、*xindex*と*yindex*は、-1にセットされる。
XRT3d_RGN_IN_LEGEND	与えられたpixel座標は、凡例の領域にある。*xindex*と*yindex*は、-1にセットされる。
XRT3d_RGN_IN_HEADER	与えられたpixel座標はグラフ領域でなく、ヘッダー領域にある。*xindex*と*yindex*は、-1にセットされる。
XRT3d_RGN_IN_FOOTER	与えられたpixel座標はグラフ領域でなく、フッター領域にある。*xindex*と*yindex*は、-1にセットされる。

Xrt3dUnpick()

*Xrt3dUnpick()*は*Xrt3dPick()*と反対の手続きで、指定されたデータ・ポイントに対するpixel座標を決定します。グリッド・データでは、xindexとyindexで参照されるグリッド・ポイントを返し、ポイント・データでは、xindex(series)とyindex(ポイント数)で指定されるスキャッター・プロット・ポイントを返します。この方法は、Xlibルーチンを利用してグラフ上に描画する場合に使用できます。詳しくは、Appendix Cをご覧ください。

```
static void
StartDrag(graph, client_data, bev)
Widget              graph;
caddr_t             client_data;
XButtonEvent        *bev;
{

    /* Start the dragging. If a valid point is selected
    * initialize the drag previewing
    */

    Xrt3dPickResult      p;
    Xrt3dData               *tmp_data

    Xrt3dPick(graph, bev->x, bev->y, &p);
    if (p.xindex != -1 || p.yindex != -1) {
        InitDrag(graph);
        XtVaSetValues(graph, XtNxrt3dSurfaceData, &tmp_data, NULL);
        printf(" You picked point (%d,%d),
            the value is %f\n", p.xindex, p.yindex,
            tmp_data->g.values[p.xindex] [p.yindex] );
    }
}
```

図44　Xrt3dPick()を使用したイベント・ハンドリング手続き

Xrt3dMap()

*Xrt3dMap()*は、XRT/3dウィジェットのハンドル、(x, y)pixel座標、map構造体のポインタを取ります。グリッド・データでは、pixelポイントの中心をサーフェス上のポイントにマップします。サーフェス値では、近くのグリッド値をinterpolation(補間)することにより取得します。ポイント・データでは、(画面座標空間で)一番近いポイントの座標を返します。

*Xrt3dMap()*は、pixel座標とグラフ座標の浮動小数点値をマップ構造体に書き込みます。

```
Xrt3dRegion
Xrt3dMap(Widget graph, Position pix_x, Position pix_y,
     Xrt3dMapResult map)
```

Xrt3dMapResultは、次のように定義されています。

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

フィールドは次の通りです。

pix_x	*Xrt3dMap()*に渡されたX座標。
pix_y	*Xrt3dMap()*に渡されたY座標。
x	マップされたグラフ座標のX値。
Y	マップされたグラフ座標のY値。
Z	マップされたグラフ座標のZ値。

Xrt3dMap() は 、 XRT3D_RGN_NOWHERE 、 XRT3D_RGN_IN_FOOTER 、 XRT3D_RGN_IN_HEADER、XRT3D_RGN_IN_LEGENDまたはXRT3D_RGN_IN_GRAPHを返します。マップされたpixelがサーフェス上にない時は、マップの結果のx、y、zは全てXRT3D_HUGE_VALにセットされます。グリッド・データでは、サーフェス上にない時pixelは無効、ポイント・データでは、(データ・セットにポイントがなく、全てのポイントが無効にならない限り)、無効なpixelはありません。

Xrt3dUnmap()

*Xrt3dUnmap()*は*Xrt3dMap()*の反対の手続きで、グラフ座標からpixel座標にマップします。Xlibルーチンを利用してグラフ上に描画する場合に使用できます。詳しくは、Appendix Cをご覧ください。

Xrt3dComputeZValue()

グリッド・セルまたはデータ・ポイントの値は、*Xrt3dComputeZValue()*を用いて変更することができます。この関数には、変更されるグリッド・セルのインデックスまたはポイントのシリーズとポイントのインデックスと、新しいポイントのpixel位置が渡されます。

*Xrt3dComputeZValue()*は、グリッド値またはデータ・ポイントのインタラクティブな変更に使用できます。例えば、ユーザがサーフェスのある箇所をクリックすると、*Xrt3dPick()*が呼び出され、一番近いグリッドのインデックスまたはポイントが決定されます。次に、マウスがドラッグされると、新しいpixel値が、グリッドまたはポイントのインデックスと一緒に*Xrt3dComputeZValue()*に渡され、そのグリッド・セルまたはポイントの新しいZ値が返されます。ドラッグが終了すると、新しいZ値はグリッド・セルまたはポイントに配置され、アップデートされたデータ構造体がXtNxrt3dSurfaceDataにセットされます。*feedback3d*のデモに、このユーザ・インタラクションの使用例がありますので参照してください。

棒グラフのマップとピック

*Xrt3dMap()*と*Xrt3dPick()*は、棒グラフでも、3Dサーフェス・グラフやスキャッター・プロットと同様に使用できます。棒以外の場所にあるpixel座標が渡された場合は、*Xrt3dMap()*はマップの結果のx, y, zエレメントに`XRT3D_HUGE_VAL`を返し、*Xrt3dPick()*はピックの結果のxとyのインデックスに-1を返します。

棒の上、下、横の部分を表示するpixelの場合は、*Xrt3dMap()*は、そのpixelの実際のX、Y、Z値を返します。

7.7 ウィンドウのリサイズ

アプリケーションの多くは、XRT/3dウィジェットを含むウィンドウのリサイズをユーザに許可し、ウィジェットを新しいウィンドウのサイズに調節することが必要です。XRT/3dウィジェットは、Motif Composite Widgetsの内部に位置することができます。通常、Composite WidgetがXRT/3dウィジェットの最適なサイズを決定し、XRT/3dはそれに従って自動的に調節されます。

エクスポースとリサイズのコールバック

アプリケーションは、`XtNxrt3dExposeCallback`、`XtNxrt3dResizeCallback`を使用してコールバックを指定することにより、XRT/3dウィジェットがいつエクスポーズまたはリサイズされたかを知ることができます。コールバックは、ウィジェットのハンドルとクライアント・データのポインタと共に呼び出されます。三番目のコールバックの引数、`call_data`は、コールバックの追加の情報を持つ`Xrt3dCallbackStruct`を指します。詳細はAppendix Aを参照してください。リサイズのコールバックは、グラフがリサイズされた後に呼び出され、エクスポーズのコールバックは、グラフが再描画された後に呼び出されます。

アプリケーションは、Xlibの機能を用いてグラフ上に描画する際に、これらのコールバックを使用できます。また、XRT/3dウィジェットのサイズに応じてウィジェットのリソースを調節する時にも、これらのコールバックが使用できます。

次のコードはリサイズのコールバックを登録します。

```
XtAddCallback(mygraph, XtNxrt3dResizeCallback,
  resize_proc, &my_data);
```

次のリサイズのコールバックは、XRT/3dウィジェットが小さくなりすぎた時に凡例を非表示にします。

```
void
resize_proc(Widget w, XtPointer client_data,
     XtPointer call_data)
{
  Dimension   width;
  Boolean     show;

  XtVaGetValues(w,
    XtNwidth,                  &width,
    XtNxrt3dLegendShow,    &show,
    NULL);

  if ( width <= 300 && show )
    XtVaSetValues(w,
    XtNxrt3dLegendShow, False,
    NULL);
  else if ( width > 300 && !show)
    XtVaSetValues(w,
      XtNxrt3dLegendShow, True,
      NULL);
}
```

8

高度なプログラミング

4Dサーフェス・グラフ ■ 4D棒グラフ

テキスト・オブジェクト ■ 分布表(Distribution Table)のカスタマイズ

凡例ラベルのカスタマイズ ■ 等高線のカスタマイズ

国際化サポート ■ MS Windowsへのポーティング

C++の使用 ■ UILの使用

リソース値をストリングに変換する

このセクションでは、XRT/3dの基本操作を習得した方へ、更に進んだ使用法を紹介します。

8.1 4Dサーフェス・グラフ

サーフェスと棒グラフでは、色で四番目の次元を表すことにより4Dグラフを作成することができます。色に関する情報は、`XtNxrt3dZoneData`リソースを用いて、二番目の`Xrt3dData`構造体としてXRT/3dに渡されます。

4Dグラフを作成するには、次のようにしてください。

- DrawZonesとDrawShadedを`True`にします。
- `XtNxrt3dSurfaceData`を使用して、3Dサーフェスを作成するようにデータを接続します。
- ゾーンと等高線の色に使用されるデータを`XtNxrt3dZoneData`として接続します。
- 両方の`Xrt3dData`構造体が、同じ原点、同じ数のXとYのポイントを持っていることを確認してください。規則的なグリッド・データが使用されている場合は、*xstep*と*ystep*は両方の構造体で同一でなければなりません。不規則的なグリッド・データが使用されている場合は、*xgrid*と*ygrid*の値が両方の構造体で同一でなければなりません。規則的なグリッド・データセットをサーフェス・データに使用する一方で、不規則なグリッド・データをゾーン・データに使用することや、またはその逆もできません。これらの条件が全て満たされないと、4Dグラフは表示されません。

ゾーン・データが、サーフェス・データにない穴(hole)を持つ場合、穴の領域のサーフェスは、ゾーン・データが渡されていないものとして表示されます。

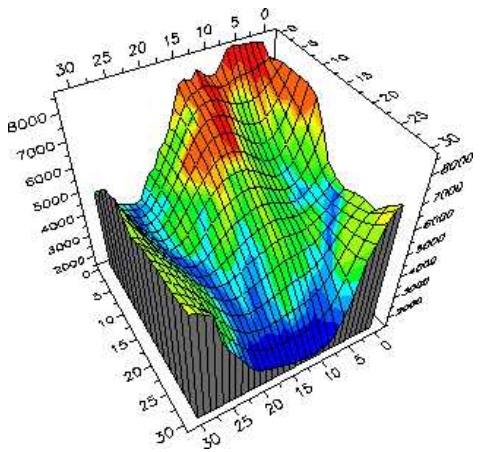


図45　4Dサーフェス・グラフ

8.2　4D棒グラフ

XtNxrt3dZoneDataが棒グラフに与えられると、ゾーン・データの値は分布表(distribution table)と一緒に使用され、棒グラフにゾーン色を適用します。

ゾーン・データが提供され、DrawZonesがTrueであると、棒は、一本が複数の色に分けられるのではなく、各棒がゾーン付けされた高さに従ってそれぞれ色付けされます。ゾーン・データが提供されている時は、等高線は描かれません。図46は4D棒グラフの例です。

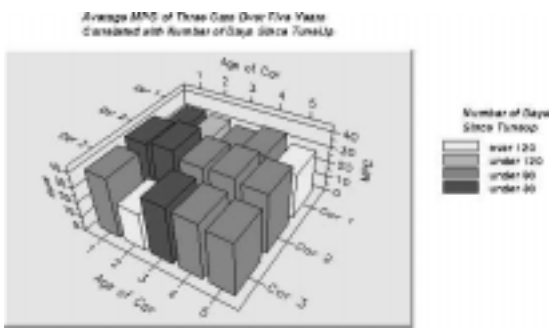


図46　4D棒グラフ

棒グラフでは、ゾーン・データの構造体は、DrawZonesがTrueの場合にのみ参照されます。凡例は、分布表(distribution table)に基づいて生成されます。凡例のラベルは、XtNxrt3dLegendStringsで置き換えたり、新たに指定することができます。凡例のストリングについては、セクション8.5をご覧ください。

8.3 テキスト・オブジェクト

サーフェスや棒グラフは、複数の*テキスト・オブジェクト(text object)*を持つことができます。テキスト・オブジェクトは、独立した矩形の領域で、次の三つの方法でグラフに貼り付けることができます。

- グリッド・インデックス
- 3D空間のポイント
- pixel座標のセット

テキスト・オブジェクトは、新しいXt Intrinsics Objectクラスである`xtXrt3dTextObject`として定義されています。Xt Objectsは、Motifのガジェットと似ていますが、geometryまたはsensitivityリソースを持ちません。テキスト・オブジェクトは、独自のリソースを持ち、ウィジェットやガジェットと同様にXtの手続きを用いて操作できます。[1]

テキスト・オブジェクトは、サーフェス上のポイントの注釈や、ユニット・キューブ内のヘッダー等、あらゆる方法に利用できます。

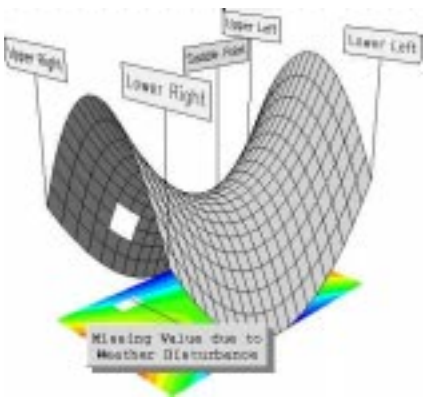


図47　サーフェスに貼り付けられたpixelとグリッドのテキスト・オブジェクト

テキスト・オブジェクトの型

XtNxrt3dTextAttachMethodリソースは、どのようにグラフにテキスト・オブジェクトを貼り付けるか指定し、テキスト・オブジェクトの型を定義します。

[1] XRT/3dは、Intrinsics Constraintクラスのサブクラスではないので、テキスト・オブジェクトでは、ウィジェットを管理するXtの手続き(*XtCreateManagedWidget()*、*XtManageChild()*等)は使用できません。テキスト・オブジェクトをmanageしようとすると、プログラムは、致命的なエラーと表示し中止してしまいます。

貼り付け法がXRT3D_ATTACH_INDEXまたはXRT3D_ATTACH_POINTの時は、Gridテキスト・オブジェクトが指定されます。(Gridテキスト・オブジェクトは、ポイント・データにも使用されます。) 貼り付け法がXRT3D_ATTACH_PIXELの時は、Pixelテキスト・オブジェクトが指定されます。2Dグラフ(DrawMeshとDrawShadedがFalseにセットされているサーフェスまたはスキャッター・プロット)では、Gridテキスト・オブジェクトは、2D Gridテキスト・オブジェクトとして参照されます。図48は、テキスト・オブジェクトに適用されるリソースです。

次は、各テキスト・オブジェクトの型の特徴です。

- **2D Gridテキスト・オブジェクト** ユニット・キューブ自身に貼り付けられ、グラフ上に描かれます。グラフと一緒にスケーリングや移動し、ストローク・フォントを使用します。
- **3D Gridテキスト・オブジェクト** ユニット・キューブ自身(X-Yのグリッド・ポイント、または3D空間のポイント)に貼り付けられ、後方から前方に描かれます。グラフと一緒に回転、スケーリング、移動し、ストローク・フォントを使用します。
- **Pixelテキスト・オブジェクト** ヘッダーやフッターと似ており、ウィジェットのpixel座標のセットに貼り付けられます。グラフ上に描かれますが、グラフが回転、スケーリング、移動しても、位置は変わりません。XmFontListフォントを使用し、枠線はカスタマイズできます。

注意: 重なり合うテキスト・オブジェクトは、正しく表示されません。アプリケーションでは、テキスト・オブジェクトが重なり合わないように確認してください。

共通のResources

XtNxrt3dTextAdjust	XtNxrt3dTextAttachMethod	XtNxrt3dTextBackgroundColor
XtNxrt3dTextForegroundColor	XtNxrt3dTextLineShow	XtNxrt3dTextOffset[XY]*
XtNxrt3dTextPrintFont	XtNxrt3dTextShow	XtNxrt3dTextStrings

*XtNxrt3dTextOffset[XY]は、3D Gridテキスト・オブジェクトには使用されない。

GridテキストオブジェクトだけのResources

XtNxrt3dTextAttachIndex[XY]	XtNxrt3dTextAttachPoint[XYZ]	XtNxrt3dTextPlane
XtNxrt3dTextStrokeFont	XtNxrt3dTextStrokeSize	

Pixelテキスト・オブジェクトだけの Resources

XtNxrt3dTextAttachPixel[XY]	XtNxrt3dTextBorder	XtNxrt3dTextBorderWidth
XtNxrt3dTextFont		

図48 テキスト・オブジェクトのリソース

テキスト・オブジェクトの作成

グラフにテキスト・オブジェクトを追加するには、次の例のようにグラフの子として作成してください。

```
Widget text_obj;
char *strings[] = {"Hello World!", NULL};
  text_obj = XtVaCreateWidget("text_obj",
  xtXrt3dTextObjectClass, graph,
  XtNxrt3dTextStrings, strings,
  ...
  NULL);
```

テキスト

テキスト・オブジェクトに表示されるテキストは、XtNxrt3dTextStringsによって指定されます。テキスト・オブジェクトには、複数行のテキストを表示できます。

複数行のテキストの行揃えは、XtNxrt3dTextAdjustで指定します。XRT3D_ADJUST_LEFT、XRT3D_ADJUST_CENTER、XRT3D_ADJUST_RIGHTは、それぞれテキストを左揃え、中央揃え、右揃えにします。

3D Gridオブジェクトの位置決め

グリッド・インデックスにテキスト・オブジェクトを貼り付けるには、XtNxrt3dTextAttachMethodをXRT3D_ATTACH_INDEXに、XtNxrt3dTextAttachIndex[XY]をグラフのXとYのインデックスにセットします。XとYの値は、データ・セットのXとYの範囲内でなければなりません。[1] テキスト・オブジェクトは、サーフェスのZ最大値の上部に表示されます。オフセットのリソースは無視されますが、テキスト・オブジェクトの方向は、XtNxrt3dTextPlaneで指定できます。

3D空間のポイントにテキスト・オブジェクトを貼り付ける場合は、XtNxrt3dTextAttachMethodをXRT3D_ATTACH_POINTに、XtNxrt3dTextAttachPoint[XYZ]を3DのX、Y、Zのポイントにセットします。これらの値は、データ範囲内でなければなりません。指定したZ値がサーフェスのZ値より小さい場合、テキスト・オブジェクトは、Z最小値の下部に表示されます。データ・セットのZ値(またはhole値)と同じかそれ以上の場合は、テキスト・オブジェクトは、Z最大値の上部に表示されます。オフセットのリソースは無視されますが、テキスト・オブジェクトの方向は、XtNxrt3dTextPlaneで指定できます。次の例は、3D Gridテキスト・オブジェクトの位置決めを示しています。

```
XtVaSetValues(my_text,
  XtNxrt3dTextAttachMethod, XRT3D_ATTACH_POINT,
  XtNxrt3dTextAttachPointX, Xrt3dDoubleToArgVal(20.0),
  XtNxrt3dTextAttachPointY, Xrt3dDoubleToArgVal(20.0),
  XtNxrt3dTextAttachPointZ, Xrt3dDoubleToArgVal(500.0),
  NULL);
```

2D Gridオブジェクトの位置決め

2Dサーフェスのテキスト・オブジェクトは、3Dサーフェスと同様にして貼り付けることができます。(2Dのポイントに貼り付けられているテキスト・オブジェクトではXtNxrt3dTextAttachPointZは無視されます。)これらは作成された順にグラフ上に描画されます。貼り付けられたポイントからテキスト・オブジェクトを移動するには、XtNxrt3dTextOffset[XY]を使用してください。

ポイント・データ・オブジェクトの位置決め

指定されたseriesとポイントにテキスト・オブジェクトを貼り付ける時は、XtNxrt3dTextAttachMethodをXRT3D_TEXT_ATTACH_INDEXにXtNxrt3dTextAttachIndexSeriesとXtNxrt3dTextAttachIndexPointを該当するインデックスにセットしてください。

---

[1] XとYインデックスのZ値がholeの場合、holeはデータの範囲外とみなされるので、テキスト・オブジェクトは表示されません。

XtNxrt3dTextAttachMethodをXRT3D_TEXT_ATTACH_POINTにセットすることにより、テキスト・オブジェクトは、3D Gridオブジェクトの場合と同様に３D空間の任意のポイントに貼り付けることができます。

Pixelオブジェクトの位置決め

ウィジェットのpixel座標にテキスト・オブジェクトを貼り付けるには、XtNxrt3dTextAttachMethod を XRT3D_ATTACH_PIXEL に、XtNxrt3dTextAttachPixel[XY]を(x, y)pixel座標のセットに設定します。これらはGridテキスト・オブジェクトの後に、作成された順に描画されます。貼り付けられたポイントからテキスト・オブジェクトを移動するには、XtNxrt3dTextOffset[XY]を使用してください。

接続線

XtNxrt3dTextLineShowを使用して、テキスト・オブジェクトと貼り付けポイントの間に線を描くことができます。このリソースは、テキスト・オブジェクトと貼り付けポイントの間の距離を設定するために、XtNxrt3dTextOffset[XY]と一緒に使用してください。

フォント

Gridテキスト・オブジェクトは、フォントの指定にXtNxrt3dTextStrokeFontとXtNxrt3dTextStrokeSizeを使用します。ストローク・フォントはAppendix Eにあります。Pixelテキスト・オブジェクトは、フォントの指定にXtNxrt3dTextFontを使用します。このリソースは、ヘッダー、フッター、凡例のフォントと同様に指定できます。

枠線

Pixelテキスト・オブジェクトの枠線は、XtNxrt3dTextBorderとXtNxrt3dTextBorderWidthで指定できます。XRT/3dで使用可能な枠線ならどれでも使用できます。(3.11参照) Gridテキスト・オブジェクトには、これらのリソースは使用されず、テキスト・オブジェクトの前景色を使用した1 pixel幅の線が描かれます。背景色がNULLの場合、枠線は描かれません。

背景のイメージ

XtNxrt3dTextImageとXtNxrt3dTextImageLayoutを使用してテキスト・オブジェクトの背景にイメージを表示することができます。通常、テキスト領域の最小サイズは、テキストのサイズによって決定されますが、XtNxrt3dTextImageMinimumSizeによって、テキスト領域の最小サイズを背景のイメージと同じサイズにすることができます。(このリソースがTrueの場合、テキストがなくてもテキスト・オブジェクトが表示されます。)

色

テキスト・オブジェクトの色はXtNxrt3dTextBackgroundとXtNxrt3dTextForegroundを使用して指定します。色の指定については、2.7を参照してください。

テキスト･オブジェクトのパフォーマンス

アプリケーションでは、複数のテキスト･オブジェクトの作成や操作をバッチ処理してください。バッチ処理については、2.6を参照してください。

テキスト･オブジェクトの削除

テキスト･オブジェクトの非表示には`XtNxrt3dTextShow`を`False`にセットし、オブジェクトの破棄には、*XtDestroyWidget()*を呼び出してください。

8.4 分布表(Distribution Table)のカスタマイズ

XRT/3dは、DrawContoursと(または)DrawZonesが`True`の場合に、デフォルトで`XtNxrt3dNumDistnLevels`とリニア分布表(distribution table)を使用します。

任意の分布表を指定するには、`XtNxrt3dDistnMethod`を`XRT3D_DISTN_FROM_TABLE`にセットし、`XtNxrt3dDistnTable`リソースを使用して、任意の分布表を提供してください。

`XtNxrt3dDistnTable`は、`Xrt3dDistnTable`構造体のポインタを指定します。

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

*nentries*は、表に幾つのエントリーがあるかをXRT/3dに伝えます。*entry*は各レベルの境界を指定するdoubleの配列です。

分布表が定義されると、*nentries*は分布レベルの数(*nlevels*)を定義し、`XtNxrt3dNumDistnLevels`は無視されます。*nlevels*の分布を表示するには、*nlevels*の等高線スタイルが必要です。また、DrawZonesが`True`の場合、最後(一番上)のゾーンに(*nlevel*+1)番目の等高線スタイルの塗りつぶし色も必要です。

例えば、次のコードは二つのレベルを指定する分布表を定義します。始めの塗りつぶし色は、≤-5の値を全て表示します。二番目のスタイルは、>-5、≤40の値を表示します。三番目のスタイルのゾーン塗りつぶし色は、>40の値を表示します。

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

mytable.nentries = XtNumber(e);
mytable.entry = e;

XtVaSetValues(mygraph,
  XtNxrt3dDistnMethod, XRT3D_DISTN_FROM_TABLE,
  XtNxrt3dDistnTable, &mytable,
  NULL);
```

指定された分布表では、デフォルトでデータによってスパンされた分布表の範囲のみが表示されます。`XtNxrt3dLegendDistnRange`が`XRT3D_DISTN_RANGE_ALL`にセットされると、分布表全体が凡例に表示されます。

8.5 凡例ラベルのカスタマイズ

任意の凡例ラベルの指定には、二つの方法があります。任意のラベリング関数を定義するか、または、分布表(distribution table)で各レベルのラベルを指定します。

凡例ラベルを作成する関数

任意の凡例ラベルを指定するには、XtNxrt3dLegendLabelFuncを、(任意に作成した)ラベルを生成する関数名にセットします。

XRT/3dは、凡例ラベルを生成する度にこの関数を呼び出します。この関数は、使用されるラベルを必ず返すようにしてください。関数の形式は次のようになります。

```
char *
MyLegendFunc(Widget graph, int level, double zmin,
      double zmax, char * label)
```

*widget*は、XRT/3dウィジェットのハンドルです。*level*はラベルが生成されるレベル(0-indexed)です。*zmin*と*zmax*は、レベルの最小と最大値を定義します。*label*は、XRT/3dがデフォルトで生成するストリングです。

デフォルトのラベルを使用するには、XtNxrt3dLegendLabelFuncをNULLにセットします。

*label*のストリングは、テキストをフィールド内で左、中央、右揃えにする、"\l"、"\c"、"\r"という特別文字を含むことができます。各メンバはこれらの文字から始まります。但し、"\"はもう一つの"\"で必ずエスケープしてください。これらの文字が指定されない場合は、左揃えになります。

次のコードは通常の凡例を作成し、レベルがZ=212の時は、このレベルに"Boiling Zone"というラベルを付けます。

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

XtVaSetValues(graph,
    XtNxrt3dLegendLabelFunc, mylabfunc,
    NULL);
```

凡例がstep表示(レベル別の表示)され、任意のラベル生成関数がNULLを返すと、XRT/3dはその凡例のエントリをスキップします。

凡例ラベルのストリング

任意の凡例ラベルは、XtNxrt3dLegendStringsリソースにストリングのリストを提供することによっても作成することができます。グリッド･データでは、これらのストリングは、分布表(distribution table)の各レベルの注釈に使用されます。この方法は、4Dの棒グラフで、同じ色の棒のグループが共通のラベルを必要とする時に便利です。

ポイント・データでは、これらのストリングは各シリーズの注釈に使用されます。実際に使用されるストリングは、XtNxrt3dLegendLabelFuncリソースを用いて、置き換えることができます。

8.6 等高線のカスタマイズ

デフォルトでは、サーフェスと棒グラフ用に100の等高線スタイルの配列を提供されています。グラフに使用される配列は、XtNxrt3dContourStylesリソースによって指定されます。ディスプレイの種類により、異なったデフォルト(カラーまたはgreyscale)が実行時に選択されます。モノクロのサーバで表示する時は、greyscaleが使用され、自動的に灰色の濃淡を調節します。次の場合は、任意の等高線スタイルを提供する必要があります。

- あるレベルの等高線スタイルを明確に指定することが重要
- 100以上のレベルを表示する
- 等高線を目立たせて区別する(XRT/3dのデフォルトの等高線スタイルは、幅が1の黒い実線)

通常、分布(distribution)のレベルに必要な数より多くの等高線スタイルを指定する方が簡単です。*nstyles*の等高線スタイルが提供され、分布のレベル数が*nlevels*である時は、XRT/3dは、次のようにして、レベル*i (0 ≤ i ≤ nlevels)*の等高線スタイルの配列のインデックスを計算します。[1]

```
round(i x (nstyles - 1)/nlevels)
```

Xrt3dContourStyle

各等高線スタイルは、(*<Xm/xrt3d_common.h>*に宣言されている)Xrt3dContourStyle構造体によって定義されます。

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

  /* DONT TOUCH THESE */
  unsigned long res1;      /* reserved */
  unsigned long res2;      /* reserved */
} Xrt3dContourStyle;
```

[1] #define Round(x) ((int)(x)+.05))として定義されています。

Xrt3dContourStyleデータ構造体は、等高線やゾーンの表示法に関する情報を持っています。フィールドは、次の通りです。

fill_color	DrawZonesがTrueの時、レベルを区分する色。色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数。(色の指定については2.7を参照)
line_color	DrawContoursがTrueの時、レベルの等高線を区分する線の色。色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数。(色の指定については2.7を参照)
line_width	DrawContoursがTrueの時、レベルの等高線を区分する線の幅。値は必ず0以上。line_widthが0であると、線は表示されない。
lpat	DrawContoursがTrueの時、レベルの等高線を区分する線のパターン。線のパターンは2Dのビュー(DrawMeshとDrawShadedが両方ともFalseでXtNxrt3dTypeがXRT3D_TYPE_SURFACE)の場合のみ有効。線のパターンは、図49にあるXRT3D_LPAT_constantsのどれかを使用する。

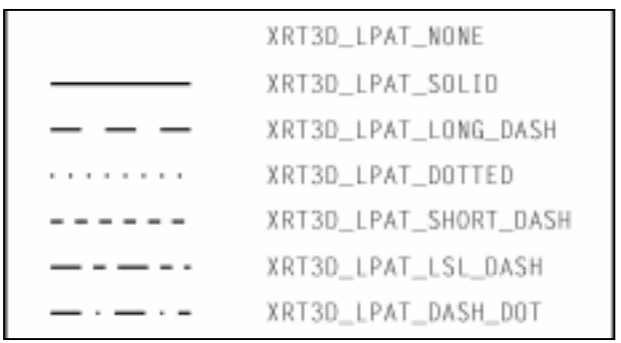


図49　線のパターン

等高線スタイルの指定

XRT/3dのデフォルトの等高線を使用したくない場合は、任意の等高線スタイルを指定することができます。

等高線スタイルは、XtNxrt3dContourStylesリソースと(または)*Xrt3dSetNthContourStyle()*と*Xrt3dGetNthContourStyle()*でアクセスできます。

XtNxrt3dContourStylesは、等高線スタイルの配列全体の取得や設定に使用できます。*XtGetValues()*と共に使用された時は、返された配列ポインタはread-onlyです。*Xrt3dDupContourStyles()*は、返された等高線スタイルの配列ポインタの複写に使用できます。

例えば、1番目と2番目の等高線スタイルの線幅を3pixelにセットする場合は、次のコードが使用できます。

```
Xrt3dContourStyle **cs, **mycs;
XtVaGetValues(myg, XtNxrt3dContourStyles, &cs,
        NULL);
mycs = Xrt3dDupContourStyles(cs);
mycs[0]->line_width = 3;
mycs[1]->line_width = 3;
XtVaSetValues(myg,
  XtNxrt3dContourStyles, mycs,
  NULL);
Xrt3dFreeContourStyles(mycs);
```

プログラムは、*Xrt3dGetNthContourStyle()*を用いて、特定の`Xrt3dContourStyles`構造体のポインタを取得することができます。次のコードは、50番目の等高線の幅をプリントし、それを二倍にしています。

```
Xrt3dContourStyle*cstyle, my_cstyle;
cstyle = Xrt3dGetNthContourStyle(myg, 49,False);
printf("line width for 50th CStyle: %d\n",
        cstyle->line_width);
my_cstyle = * cstyle;
my_cstyle.line_width = my_cstyle.line_width * 2;
Xrt3dSetNthContourStyle(myg, 49,
        &my_cstyle, False);
```

等高線スタイルについて

等高線スタイルをデフォルトに戻すには、*Xrt3dResetContourStyles()*を呼び出します。

ゾーンまたは等高線のレベルの数(*nlevels*)は、`XtNxrt3dNumDistnLevels`によって定義されます。(分布表(distribution table)が提供されている場合は、*nlevels*は表にあるエントリー数です。) DrawContoursが`True`、DrawZonesが`False`の場合は、*nlevels*の等高線スタイルが必要です。DrawZonesが`True`であれば、(*nlevels+1*)番目の等高線スタイルが最後(一番高い)のゾーンの塗りつぶし色の定義に使用されるので、*nlevels+1*の等高線スタイルが必要です。

提供された等高線スタイルを、表示される各レベルのスタイルに一つずつ使用する場合は、必ず*nlevels+1*の等高線スタイルを提供してください。

等高線スタイルの数がレベルの数より少ない場合、XRT/3dは、提供されている等高線スタイルを再利用します。

8.7 国際化サポート

Xサーバでサポートされ ているX Windowフォントであれば、どれでもヘッダー、フッター、凡例に使用することができます。漢字等の2-byteフォントもヘッダー、フッター、凡例に使用することができます。

印刷

*Xrt3dDrawPS()*は、プログラムがCロケールで実行されている場合は、ISO Latin 1文字セットを使用したPostScriptを生成します。また、この手続きは、プログラムが日本語のロケールで実行されていれば、日本語PostScriptを生成します。

8.8 MS Windowsへのポーティング

KL Group社のOlectra Chartという製品を使用して、XRT/3dのアプリケーションは、Microsoft Windowsにポーティングすることができます。Olectra Chartを使用することにより、グラフ・アプリケーションは、ネイティブのWindowsツールキットを使用し、Windows 3.1、Windows 95、Windows NTに対応することができます。このセクションでは、XRT/3dのクロス・プラットフォームAPIや、二つのプラットフォームの違いの処理の方法等を解説します。

概要

Windowsツールキットには、Motifと同様、定義済みのコントロールがあります。Olectra Chartの16と32 bitの”3D Chart”DLLカスタム・コントロールは、グラフを作成する”ウィジェット”(コントロール)をWindowsツールキットに追加します。このコントロールは、XRT/3dと同じリソース(Windowsの場合はプロパティ)、データ型、方法を持ちます。[1] 共通のAPIを使用することにより、XRT/3dアプリケーションをリコンパイルし、短時間でWindowsにポーティングさせることができます。アプリケーションは、MotifまたはWindows特有のコードを処理する場合、`#ifdef`ステートメントを使用することができます。

C++ラッパーを使用すると、アプリケーション・コードのプラットフォームへの依存が少なくなります。XRT/3dとOlectra Chartには、どちらも、C++ラッパーが提供されています。

[1] Olectra ChartにはOCXも提供されていますが、DLLのみがXRT/3dとのクロス・プラットフォームAPIを共有します。

グラフ･コントロールの作成

Windowsのプログラムでは、コントロールは異なる方法で作成されます。Motifウィジェットは、他のウィジェットの子として作成(instantiate)され、サイズや位置は作成時に定義されますが、Windowsでは、通常、コントロールは他のコントロールとは独立して、リソースまたは*.RC*ファイルで定義されます。

3D Chartコントロールを定義する時に使用されるクラス名は、"OlectraChart3D"(32-bit DLL)または"OlectraChart3D16"(16-bit DLL)です。

コントロールが一度定義された後は、Windowsは、プログラムの初期化中にコントロールを作成することができます。(XRT/3dをtoplevelシェルの子として作成する)*XtCreateManagedWidget()*の代りに、Olectra Chartのプログラムは次のようなコードを使用します。

```
static HXRT3D graph;
static HWND hwndChart;

hwndChart = GetDlgItem(hWnd, IDCHART);/* defined in .RC file */
graph = Xrt3dCreate();
Xrt3dAttachWindow(graph, hwndChart);/* joins chart and window /*
```

一つのWidgetハンドルに対して、Windowsプログラムは3D Chartコントロール(HXRT3D)と画面上のウィンドウ(HWND)の二つのハンドルが必要です。

リソースの設定と取得

XRT/3dとOlectra Chartは、リソース値の設定と取得に共通のAPIを使用します。*Xrt3dSetValues()*は*XtVaSetValues()*と同様に、一つまたは複数のリソースを設定することができます。*Xrt3dGetValues()*は*XtVaGetValues()*と同様に、リソースの現在値を取得します。*XtVaSetValues()*を呼び出してグラフのリソースをセットするMotifのコードは、*Xrt3dSetValues()*を呼ぶように変更することができます。(注意: double型のリソースにアクセスする前は、*Xrt3dDoubleToArgVal()*または*Xrt3dArgValToDouble()*を呼び出さないでください。)

各XRT/3dのリソースは、MotifとWindowsの3D Chartコードで使用できる、クロス･プラットフォーム用の名前を持っています。図50は、幾つかのリソースの例です。全リストの一覧は、P205にあります。

XRT3D_AXIS_STROKE_FONT	XRT3D_BACKGROUND_COLOR	XRT3D_DRAW_MESH	XRT3D_DRAW_ZONES
XRT3D_FOOTER_BORDER	XRT3D_FOOTER_STRINGS	XRT3D_GRAPH_HEIGHT	XRT3D_HEADER_FONT
XRT3D_LEGEND_SHOW	XRT3D_LEGEND_X	XRT3D_SURFACE_DATA	XRT3D_TYPE
XRT3D_XANNO_METHOD	XRT3D_YDATA_LABELS	XRT3D_ZMAX	XRT3D_ZORIGIN

図50　クロス･プラットフォーム対応リソース名

次の例は、(my_data変数が正しく定義されていることを前提として)三つのグラフ･プロパティをセットしています。

```
Xrt3dSetValues(graph,
  XRT3D_SURFACE_DATA,      my_data,
  XRT3D_TYPE,          XRT3D_TYPE_BAR,
  XRT3D_DRAW_ZONES,        TRUE,
  NULL);
```

ストリングを使用したリソース設定

*Xrt3dSetPropString()*と*Xrt3dGetPropString()*を使用して、アプリケーションのポーティング作業を削減することもできます。これらの手続きは、ストリング・ポインタとしてリソース値を設定/取得しますので、例えば、WindowsのCOLORREFを使用するようコードを書き直さずに、色を設定することができます。値のフォーマットは、XRT/3dのリソース・ファイルの場合と同じです。(リソース・ファイルの構文はAppendix Gにあります。)次の例では、ヘッダーの背景色をセットしています。

```
Xrt3dSetPropString(graph,
  XRT3D_HEADER_BACKGROUND_COLOR, "orange");
```

データ型のハンドリング

XRT/3dとOlectra Chart 3D Chartのプログラミングで使用されるデータ型は、ほとんど同じです。但し、二つのプラット・フォームの違いから、多少の相違点があります。XRT/3dとOlectra Chartには、これらのデータ型に対して、次のクロス・プラットフォームtypedefsが提供されています。

ポーティング可能なtypedef	XRT/3dのデータ型	Olectra Chartのデータ型
	Widget	HXRT3D
XrtBoolean	Boolean	BOOL
XrtColor	char *	COLORREF
XrtDimension	Dimension	int
XrtFont	XmFontList	HFONT
XrtPosition	Position	int

8.9 C++の使用

XRT/3dには、C++のインターフェースが提供されています。これを使用するには、ヘッダー・ファイル*$XRTHOME/include/Xm/Xrt3d.hxx*をC++プログラムにインクルードしてください。このファイルは、次の三つのクラスを作成します。

クラス	方法
Xrt3d	■ *XtCreateXrt3d()*を呼ぶconstructor。 ■ *XtDestroyWidget()*を呼ぶdestructor。 ■ XRT/3dを最初の引数として取る各Cルーチンに対応する方法。(例: *Xrt3dDrawPS()*の方法は*drawPS()*) ■ 各設定可能なリソースをセットする方法。(例: `XtNxrtType`をセットする方法は*setType()*) ■ 各リソースを取得する方法。(例: `XtNxrtBackgroundColor`を取得する方法は*getBackgroundColor()*) ■ 一般的なXtのファンクションを実行する方法。(*addCallback()*、*destroy()*、*getDisplay()*、*getName()*、*getParent()*、*isManaged()*、*isRealized()*、*manageChild()*)
Xrt3dDataC	■ *Xrt3dMakeGridData()*を呼ぶconstructor。 *Xrt3dMakeIrGridData()*を呼ぶconstructor。 *Xrt3dMakePointData()*を呼ぶconstructor。 ■ *Xrt3dMakeDataFromFile()*を呼ぶconstructor。 ■ *Xrt3dDestroyData()*を呼ぶdestructor。 ■ *Xrt3dDataCopy()*を実行するOverloaded = operator ■ (`Xrt3dData *`)のcastingサポート。 ■ 各内部データ構造体メンバの方法。
Xrt3dText	■ *XtCreateXrt3dText()*を呼ぶconstructor。 ■ *XtDestroyWidget()*を呼ぶdestructor。 ■ 各設定可能なリソースをセットする方法。(例: `XtNxrtTextAdjust`をセットする方法は*setTextAdjust()*) ■ 各リソースを取得する方法。(例: `XtNxrtTextBorder`を取得する方法は*getTextBorder()*)

簡単なC++の例が、*$XRTHOME/src/3d/examples/simple_cpp.cxx*にあります。

8.10 UILの使用

XRT/3dウィジェットの作成には、OSFのUser Interface Language(UIL)を使用することができます。UILは、標準のMotifウィジェットに使用されていないデータ型はサポートしていないので、XRT/3dのリソースの中にはUILでセットできないものもあります。但し、これらは、`MrmNcreateCallback`を用いればアプリケーションでセットすることが可能です。尚、UILの引数とデータ型の解説は、Appendix Hにあります。

UILでグラフを作成するには、次の手順に従ってください。

- XRT/3dのUILヘッダー・ファイル、*Xrt3d.uil*をインクルードする。
- *MrmInitialize()*を呼び出した直後に、*Xrt3dMrmInitialize()*を呼ぶ。

*$XRTHOME/src/3d/examples*には、UILのサンプル・プログラム(*simple_uil.c*と*simple_uil.uil*)があります。

8.11 リソース値をストリングに変換する

リソース変換プログラムは、リソース値のフォーマットを変換します。このプログラムは、通常、リソース・ファイルにある値をストリングから、ウィジェットが理解できるフォーマットに変換する場合に使用されます。(例: Booleanと整数) XRT/3dは、装備されたこのプログラムを自動的に登録し、ストリングをXRT/3d固有のリソース型(`XtRXrt3dAdjust`や`XtRXrt3dAlign`等)に変換します。

XRT/3dには、XRT/3d固有のリソース型からストリングに変換する"逆変換プログラム"も提供されています。リソース値をリソース・ファイルに書く時や、ウィジェットの状態を保管する時に便利です。

逆変換プログラムを使用するには、*Xrt3dSetReverseConverters()*を呼び出してください。各プログラムは、変換の結果の格納に必要なスペースを割り当てます。このストリングは、使用後、必ず解放してください。

次のコードは、凡例の方向の値をプリントする時、これらの変換プログラムがどのように使用できるかを示します。

```
void
print_legend_orientation()
{
 Xrt3dAlign   align;
 XrmValue     from, to;

 Xrt3dSetReverseConverters();

 XtVaGetValues(graph,
     XtNxrt3dLegendOrientation, &align, NULL);
 from.addr = (caddr_t) &align;
 from.size = sizeof(align);
 to.addr = NULL;
 if (XtConvertAndStore(graph, XtRXrt3dAlign, &from,
     XtRString, &to)) {
    printf("Legend Orientation: %s\n", to.addr);
  XtFree(to.addr);
 }
}
```

または、Xrt3dGetPropString()を使用することもできます。

```
void
print_legend_orientation()
{
 String  string;
 if (Xrt3dGetPropString(graph,
         XRT3D_LEGEND_ORIENTATION, &string))
{
 printf("Legend Orientation: %s\n", string);
 Xrt3dFreePropString(string);
 }
}
```

9

PDS ユーティリテイ・ライブラリ

XrtPDSImage ■ Image 型の登録
ダイナミックまたは共用ライブラリの使用
XrtPDSListオブジェクト ■ 手続きと方法

9.1 XrtPDSImage

XrtPDSImageは、イメージと一緒にファイル名や、幅、高さなどのプロパティを持つオブジェクトです。このオブジェクト内部では、Pixmapフォーマットが使用されています。

XRTPDSImageには、ファイル名や、幅、高さ、clip mask、Pixmap自身を返すパブリック・ファンクションがありますが、これらの属性はセットすることはできません。

XrtPDSImageを利用するとことにより、XRTウィジェットは、イメージ・オブジェクトを処理するだけでよくなり、そのイメージがどこにあるか、元のフォーマットが何であるかといったことを気にする必要がなくなります。また、新しいイメージの型も、ウィジェットのライブラリを変更することなしに追加することができます。

9.1.1 XrtPDSImageの作成

PixmapからXrtPDSImageを作成するには、*XrtPDSImageCreate()*を呼び出してください。

```
image = XrtPDSImageCreate(pixmap, display, width, height);
```

pixmapは、指定されたサイズに変更されたり切り取られたすることがないので、この場合の*width*と*height*は、pixmapの実際の幅と高さにしてください。

通常、XrtPDSImageはPixmapからユーザによって直接作成されることはありません。イメージはサポートされるフォーマットを使用してファイルに記憶されます。XRT/3dでは、ウィジェットに登録されているコンバータを使用してXrtPDSImageが作成されます。デフォルトでは、これらのコンバータは、XPM、XWD、GIFファイルをXrtPDSImageに変換することができます。PNGとJPEGを使用するには、ウィジェットにこれらのイメージ型を登録し、明示的に使用可能になるよう設定してください。尚、これ以外のイメージ型を使用する場合は、9.2.2を参照してください。

9.2 Image 型の登録

XrtPDSImageオブジェクトを使用することができるXRTウィジェットは、登録されたイメージ型のリストを持っています。登録されていない型や新しい型は、次のファンクションを使用して登録できます。

```
Boolean
Xrt3dEnableImageType(
  XtPointer    class,
  XtPointer    functions
  )
```

class はウィジェット・クラス(i.e. `xtXrt3dGraphWidgetClass`)、*function* は`XrtPDSImageFuncs`構造体へのポインタです。この構造体は次のように定義されます。

```
typedef struct {
  XrtPDSImageNameCompareProc  check_name;
  XrtPDSImageHeaderCompareProccheck_header;
  XrtPDSImageCreateProc       create_image;
  XrtPDSImageType             type;
  XrtPDSImageOutputProc       output;
} XrtPDSImageFuncs
```

例えば、JPEGの入出力は、次のようにして登録することができます。

```
#include <Xm/Xrt3d.h>
#include <Xm/XrtPDSJpeg.h>
Xrt3dEnableImageType((XtPointer)xtXrt3dWidgetClass,
  (XtPointer)XrtPDSJpegImageUtils);
```

*XrtPDSJpegImageUtils*は、定義済みの`XrtPDSImageFuncs`構造体のポインタです。この後、*libxrtjpeg.a*をアプリケーションにリンクしてください。

イメージ型が登録されると、この型は、ウイジェットが持つイメージ型のリストに追加されます。ファイルからイメージをロードするように要求が出されると、ウィジェットは、`XrtPDSImageFuncs.check_name`を呼び出してこのリストを検索し、このファイル名の型がサポートされているかを調べます。失敗時には、`XrtPDSImageFuncs.check_header`が呼び出され、ファイルの最初の250byteからファイルのイメージ型が識別できるかどうか調べます。

ファイルのイメージ型をサポートする`XrtPDSImageFuncs`構造体が検索されると、`XrtPDSImageFuncs.create_image`が呼び出され、そのイメージ・ファイルからXrtPDSImageが作成されます。

注意: `XrtPDSImageFuncs`構造体の型は、この構造体でサポートされているイメージの型を表すintです。

9.2.1 定義済みのイメージ

使用できる定義済みのイメージは次の通りです。

```
#define XRTPDS_XPM_IMAGE   0
#define XRTPDS_XWD_IMAGE   1
#define XRTPDS_GIF_IMAGE   2
#define XRTPDS_JPEG_IMAGE 3
#define XRTPDS_PNG_IMAGE   4
```

デフォルトでは、XPM、XWD、GIFファイルの読み込みと、XPMとXWDファイルの書き込みが、XRT/graphとXRT/3dでサポートされています。PNGとJPEGの入出力は、該当するライブラリをリンクしイメージ型を登録することによってサポートされるようになります。JPEGライブラリは*libxrtjpeg.a*、PNGライブラリは*libxrtpng.a*です(*libz.a*圧縮ライブラリも必要です)。

イメージのサポートは、イメージ・オブジェクトと`XrtPDSImageFuncs`構造体の二つで行われます。イメージ・オブジェクトは変換されたイメージを持ち、`XrtPDSImageFuncs`構造体は、イメージの読み書きを処理する方法を持ちます。

各イメージ・フォーマットは、`XrtPDSImageFuncs`構造体を持ちます。(例: XPMは`XrtPDSXpmImageUtil`、JPEGは`XrtPDSJpegImageUtil`) 例えば、JPEGイメージからXrtPDSImageを作成する時は、`XrtPDSJpegImageUtils.create_image`を呼び出すと、変換されたイメージを持つオブジェクトが返されます。

9.2.2 任意のイメージ

任意のイメージ・フォーマットの読み書きは、`XrtPDSImageFuncs`に該当するフィールドを設定することにより、サポートすることができるようになります。下記は、構造体とそのフィールドの例です。

```
typedef struct {
  XrtPDSImageNameCompareProc    check_name;
  XrtPDSImageHeaderCompareProccheck_header;
  XrtPDSImageCreateProc         create_image;
  XrtPDSImageType               type;
  XrtPDSImageOutputProc         output;
} XrtPDSImageFuncs;
```

例えば、Targa(.TGA)ファイルを使用する場合は、下記のように定義を作成し、ファンクションを書く必要があります。

```
/* Use a number greater than the maximum predefined type */
#define TARGA_IMAGE_TYPE 100

Boolean
CheckIsTargaFilename(
 string  filename;
 )

Boolean
CheckTargaMagicNumber(
 char    *buffer
 )

XrtPDSImage
CreateImageFromTargaFile(
 Widget  w,
 Ximage  *image,
 String  filename,
 Boolean color,
 String  msg
 )

Boolean
WriteImageToTargaFile(
 Widget  w,
 Ximage  *image,
 String  filename,
 Boolean color,
 String  msg
 )
```

次にXrtPDSImageFuncs構造体を作成し、セットしてください。

```
XrtPDSImageFuncs TargaImageUtilStruct = {
 (XrtPDSImageNameCompareProc) CheckIsTargaFilename,
 (XrtPDSImageHeaderCompareProc) CheckTargaMagicNumber,
 (XrtPDSImageCreateProc) CreateImageFromTargaFile,
 (XrtPDSImageType) TARGA_IMAGE_TYPE,
 (XrtPDSImage) WriteImageToTargaFile
};
XrtPDSImageFuncs *TargaImageUtils = &TargaImageUtilStruct
```

最後に、この新しいイメージ型を登録します。

```
Xrt3dEnableImageType(xtXrt3dWidgetClass, TargaImageUtils)
```

これで、(XtNxrtDataAreaImage等の)リソースのTargaイメージを読み込むことができるようになりました。また、Targaイメージは、*XrtSaveImageGeneric()*を呼び出すことによって書き込むこともできます。

```
Xrt3dSaveImageGeneric(graph, TARGA_IMAGE_TYPE, "output.tga", NULL)
```

9.2.3 イメージ・ファンクションの書き込み

XrtPDSImageNameCompareProc

```
Boolean
CheckIsTargaFilename(
 String      file_name
 )
```

このファンクションには、イメージのファイル名が渡されます。ファイルの名前や拡張子が有効であればTrue、そうでなければFalseを返します。

XrtPDSImageHeaderCompareProc

```
Boolean
CheckTargaMagicNumber(
 char *buffer
 )
```

このファンクションには、イメージ・ファイルの最初の250byteを持つバッファが渡され、このファイルが有効であるかどうかが調べられます。通常は、ファイルの最初の部分にあるマジック・ナンバが調べられます。(例: GIFの場合は"GIF87a"、"GIF89a") ファイルが有効であればTrue、そうでなければFalseが返されます。

XrtPDSImageCreateProc

```
XrtPDSImage
CreateImageFromTargaFile(
 Widget      w,
 String      file_name
 )
```

このファンクションには、ウィジェットとイメージ・ファイルの名前が渡され、このファイルからXrtPDSImageが作成されます。通常、この作業は次のように行われます。

1 イメージをネイティブ・フォーマットでロードする
2 Pixmapに変換する
(Xpmライブラリにあるファンクションを利用すると便利です。)
3 *XrtPDSImageCreate()*でこのpixmapよりXrtPDSImageを作成する

XrtPDSImageOutputProc

```
Boolean
WriteImageToTargaFile(
 Widget  w,
 Ximage  *image,
 String  filename,
 Boolean color,
 String  msg
 )
```

このファンクションには、ウィジェット、変換され保管されるXimage、イメージが保管されるファイルの名前、白黒かカラーかを示すBoolean、エラー・メッセージ用のメッセージ・バッファが渡されます。バッファがNULLであれば、エラー・メッセージは、stdoutまたはstderrにプリントされます。

9.3 ダイナミックまたは共用ライブラリの使用

JPEGとPNGイメージ・ライブラリは、スタティック(静的)ライブラリとして提供されています。ダイナミック・ライブラリや共用ライブラリを作成する時は、(インターネットから無料でダウンロードできる)ソースコードを取得し、リコンパイルする必要があります。各イメージ・ライブラリには、カスタマイズされたXRTモジュールが追加されていますが、それ以外は変更されていません。このモジュールは、共用ライブラリに簡単に追加できるよう、コンパイルされた形で提供されています。（このモジュールは、*$XRTHOME/lib-shared*にあります。）

このモジュールは、XRT PDSの共用ライブラリが用意されている機種にのみ提供されています。対応機種に関しては、リリースノートを参照してください。

9.4 XrtPDSListオブジェクト

XrtPDSListオブジェクトは、同じサイズのアイテムのグループを記憶させる時に便利なオブジェクトです。提供されている方法を使用して、これらのアイテムの追加、削除、並べ替え、取得等を簡単に行うことができます。

XrtPDSListオブジェクトで指定されるリストは0ベースです(最初のエレメントのインデックスは0、二番目のエレメントのインデックスは1....というように続きます)。

9.4.1 サンプル・コード

下記は、リストを作成し、アイテムを追加、そして取得している例です。

```
void
CreateListOfIntegers()
{
 XrtPDSList list;
 int i, *ptr;

 list = XrtPDSListCreate(sizeof(int));
 i = 34;
 XrtPDSListAppend(list, &i);
 i = 46;
 XrtPDSListAppend(list, &i);
 ptr = XrtPDSListGetItem(list, 1);
 fprintf (stderr, "item is %d\n", *ptr);
}
```

この手続きでは整数のリストを作成し、整数34と46を追加しています。次にリストの二番目のアイテムを取得しそれを表示しています。

9.4.2 リストの作成と破棄

リストを作成するには、*XrtPDSListCreate()*を呼び出してください。

```
list = XrtPDSListCreate(sizeof(int));
```

この方法には、リストの各エレメントのサイズを表すパラメータが渡されます。(異なる型のエレメントは、サイズが同じであれば同じリストに記憶させることができます。)この方法は、作成されたリストを返します。

*XrtPDSListCreate()*で作成されたリストを破棄するには、*XrtPDSListDestroy()*を呼び出してください。

```
XrtPDSListDestroy(list);
```

リストで使用されていた全てのメモリは解放されます。

リストのコピー

既存のリストのコピーを作成するには、*XrtPDSListDuplicate()*を使用します。

```
newlist = XrtPDSListDuplicate(oldlist);
```

これはリストとそのエレメントのコピーを作成し、コピーのポインタを返します。元のリストは変更されません。

9.4.3 リストのアイテムの追加と削除

アイテムの追加

アイテムは、既存のリストの次の四つの場所に追加することができます。

- リストの最後尾
- 既存のアイテムの直前
- 既存のアイテムの直後
- ソートされた順序での該当個所

リストの最後尾にアイテムを追加するには、*XrtPDSListAppend()*を呼び出してください。

```
index = XrtPDSListAppend(list, newitem);
```

この方法は、追加の対象となるリストと追加されるアイテムの二つのパラメータが必要です。アイテムのサイズは、リストが*XrtPDSListCreate()*によって作成される時に指定されたサイズと同じにしてください。この方法は、新しいアイテムのインデックスを返します。(インデックス0は最初のエレメント、1は二番目のエレメントと、続きます。)

既存のアイテムの前に新しいアイテムを追加するには、*XrtPDSListInsetBefore()*を呼び出します。

```
newindex = XrtPDSListInsertBefore(list, oldindex, newitem);
```

*XrtPDSListInsertBefore()*は、追加の対象となるリスト、既存のアイテムのインデックスと追加されるアイテムの三つのパラメータが必要です。この方法は、新しいアイテムのインデックスを返します。同様にして、*XrtPDSListInsertAfter()*は、既存のアイテムの直後に新しいアイテムを追加します。

```
newindex = XrtPDSListInsertAfter(list, oldindex, newitem);
```

この方法のパラメータと戻り値は、*XrtPDSListInsertBefore()*と同じです。

リストがソートされている場合は、*XrtPDSListAdd()*によって該当位置にアイテムを追加します。

```
index = XrtPDSListAdd(list, newitem);
```

この方法は、追加の対象となるリストと追加されるアイテムの二つのパラメータを取ります。新しいアイテムのインデックスが戻ります。(詳細は9.4.6を参照してください。)

アイテムの除去と削除

XrtPDSListオブジェクトでは、アイテムをリストから除去(remove)したり、削除(delete)したりすることができます。

リストからアイテムを除去するには*XrtPDSListRemove()*を呼び出してください。

```
retpos = XrtPDSListRemove(list, position);
```

この方法には、リストと除去されるアイテムの位置が渡されます。リストから全てのアイテムを除去するには、*XrtPDSListRemoveAll()*を呼びます。

```
XrtPDSListRemoveAll(list);
```

上の二つの方法では、アイテムは除去されているだけでメモリは残っています。アイテムを除去すると同時に破棄するには、*XrtPDSListDelete()*を呼び出してください。

```
retpos = XrtPDSListDestroy(list, position);
```

この方法には、*XrtPDSListRemove()*と同様に、リストと除去されるアイテムの位置が渡されます。リストの全てのアイテムを破棄するには、*XrtPDSListDestroyAll()*を呼び出してください。

```
XrtPDSListDestroyAll();
```

特定なアイテムに関連したデータ構造体が定義されており、アイテムが削除される時にその構造体も破棄するには、*XrtPDSListSetDestroyProc()*を呼び出し、破棄の手続きを定義してください。

```
XrtPDSListSetDestroyProc(list, my_destroy_proc);
```

この手続きは、*XrtPDSListDestroy()*または*XrtPDSListDestroyAll()*がコールされる度に呼び出されます。この手続きは、次のように定義してください。

```
void my_destroy_proc (
 XrtPDSList   object,
 XtPointer    item
 );
```

*object*はリスト、*item*は破棄されるアイテムです。

XrtPDSListオブジェクトでは、リスト上のオブジェクトを*XrtPDSListDestroyObject()*によって破棄することもできます。

```
XrtPDSList list, *objptr;

XrtPDSListDestroyObject(list, objptr);
```

ここの二つのパラメータはリストと破棄されるオブジェクトのポインタです。

リスト上のオブジェクトではないアイテムを破棄するには、*XrtPDSListDestroyXtPointer()*を呼び出してください。

```
XrtPDSList list;
XtPointer *item;

XrtPDSListDestroyXtPointer(list, item);
```

この方法には、リストと破棄されるアイテムのポインタが渡されます。

9.4.4 アイテムの検索と取得

XrtPDSListオブジェクトは、下記の検索と取得を操作する方法を提供しています。

- XrtPDSListがリストであるかを調べる
- リスト上でアイテムを検索する
- インデックス付きのエレメントまたはエレメントの配列全体を取得する
- リスト上を前方向または後ろ方向に検索する

その他の方法では、アイテムの数やそのサイズを取得することができます。

リストのチェック

*XrtPDSListIsList()*は、オブジェクトがXrtPDSListであるかどうかを調べます。

```
Boolean       is_list;
XrtPDSList    object;

is_list = XrtPDSListIsList(object);
```

*XrtPDSListISList()*は、オブジェクトがXrtPDSListオブジェクトであれば`True`を、そうでなければ`False`を返します。

アイテムを検索する

リスト上のアイテムを探し出すには、次の方法を使用してください。

- XrtPDSListの比較手続きを使用して、指定されたアイテムに適合するアイテムを探し出す*XrtPDSListFind()*
- 対応するメモリを比較しながら、指定されたアイテムと適合するアイテムを探し出す*XrtPDSListLookupItem()*

どちらの方法にも、リストと比較されるアイテムが渡されます。また、この二つの方法は、適合したアイテムがみつかった場合はそのインデックス、みつからなかった場合は`XRTPDS_LIST_ITEM_NOT_FOUND`を返します。

*XrtPDSListFind()*によって使用される比較手続きを指定するには、*XrtPDSListSetCompareProc()*を呼び出してください。

```
XrtPDSListSetCompareProc(list, my_compare_proc);
```

この比較手続きは、次のプロトタイプを持つように定義してください。

```
int my_compare_proc (
 XtPointer   item1,
 XtPointer   item2
 );
```

*item1*と*item2*は、比較されるアイテムです。この手続きは、下記を返します。

- *item1*が*item2*より大きい場合は正の整数
- 二つのアイテムが同じであれば0
- *item1*が*item2*より小さければ負の整数

現在使用されている比較手続きを取得するには、*XrtPDSListGetCompareProc()*を使用してください。

```
XrtPDSListItemCompareProc myproc;

myproc = XrtPDSListGetCompareProc(list);
```

返された手続きは、`XrtPDSListItemCompareProc`型で、次のように定義されます。

```
typedef int (*XrtPDSListItemCompareProc) (
XtPointer item1, XtPointer item2
);
```

この比較手続きは、並べ替えにも使用できます。詳細は9.4.6を参照してください。

インデックスによってアイテムを取得する

位置によってアイテムを取り出すには、*XrtPDSListGetItem()*を呼び出します。

```
XtPointer     item;
XrtPDSList    list;
Int           index;

item = XrtPDSListGetItem(list, index);
```

*XrtPDSListGetItem()*には、リストと返されるアイテムの位置(0は一番目のアイテム、1は二番目のアイテム…)の二つのパラメータが渡されます。この位置のアイテムが返されます。位置が無効であれば、`NULL`が返ります。

特定な位置でのアイテムをセットするには、*XrtPDSListSetItem()*を呼び出します。

```
XrtPDSListSetItem(list, index, item);
```

この方法には、リスト、新しいアイテムのインデックスと新しいアイテム自身の三つのパラメータが渡されます。この方法は、セットが成功すれば`True`、指定されたインデックスが無効であれば`False`を返します。

リスト配列の取得

リスト・エレメントの配列全体を取得するには、*XrtPDSListGetItems()*を呼び出します。

```
XrtPDSList    list;
XtPointer     array;

array = XrtPDSListGetItems(list);
```

リストが空であれば、NULLが返ります。

アイテムの検索

指定されたアイテムを前方向に検索するには、*XrtPDSListSearchForward()*を呼び出します。

```
int           pos;
XrtPDSList    list;
int           start_position;
XtPointer     item;

pos = XrtPDSListSearchForward(list, start_position, item);
```

この方法は、リスト、検索の開始される位置、検索されるアイテムの三つのパラメータを必要とします。アイテムが適合すれば、そのアイテムの位置が返りますが、適合しなければXRTPDS_LIST_ITEM_NOT_FOUNDが返ります。リストを後方向に検索するには、*XrtPDSListSearchBackward()*を呼び出します。

```
pos = XrtPDSListSearchBackward(list, start_position, item);
```

パラメータと戻り値は、*XrtPDSListSearchForward()*の場合と同じです。

リスト情報を取得する

リスト上のアイテム数を取得するには、*XrtPDSListGetItemCount()*を呼び出します。

```
int           numitems;
XrtPDSList    list;

numitems = XrtPDSListGetItemCount(list);
```

この方法には、リストが引数として渡されます。

アイテムのサイズを取得するには、*XrtPDSListGetItemSize()*を呼び出してください。

```
int           itemsize;
XrtPDSList    list;

itemsize = XrtPDSListGetItemSize(list);
```

9.4.5 リスト比較ルーチン

XrtPDSListオブジェクトは、異なる型の値の比較に使用できるルーチンを幾つか提供しています。倍精度の浮動小数点値、単精度の浮動小数点値、整数、ストリングを比較するルーチンが定義されています。これらは全て、*XrtPDSListFind()*で使用することができます。その場合は、*XrtPDSListSetCompareProc()*に使用するルーチンを渡してください。

二つの倍精度浮動小数点値を比較するには、*XrtPDSCmpDoubles()*を呼び出してください。

```
XtPointer     double1, double2;
int           retval;

retval = XrtPDSCmpDoubles(double1, double2);
```

このルーチンは、一番目の値が二番目の値より大きい時は1、同じであれば0、小さければ-1を返します。

*XrtPDSCmpFloats()*は、二つの単精度浮動小数点値を比較します。

```
XtPointer     float1, float2;
int           retval;

retval = XrtPDSCmpFloats(float1, float2);
```

戻り値は、*XrtPDSCmpDoubles()*の場合と同じです。

二つの整数を比較する場合は、*XrtPDSCmpInts()*を使用してください。

```
XtPointer     int1, int2;
int           retval;

retval = XrtPDSCmpInts(int1, int2);
```

戻り値は、*XrtPDSCmpDoubles()*の場合と同じです。

二つのストリングを比較するには、*XrtPDSCmpStrings()*を呼び出してください。

```
XtPointer     string1, string2;
int           retval;

retval = XrtPDSCmpStrings(string1, string2);
```

このルーチンは、一番目のストリングが二番目のストリングよりアルファベット順で後に来る時は1、同じであれば0、先に来る場合は-1を返します。

*XrtPDSCmpStrings()*は大文字と小文字を区別して扱います。大文字・小文字を区別しないで比較を行うには、*XrtPDSCmpStringsIgnoreCase()*を使用してください。

```
XtPointer     string1, string2;
Int           retval;

retval = XrtPDSCmpStringsIgnoreCase(string1, string2);
```

戻り値は、*XrtPDSCmpStrings()*の場合と同じです。

9.4.6 リストのソート

リスト上のアイテムをソートするには、*XrtPDSListSort()*を呼び出してください。

```
XrtPDSListSort(list);
```

*XrtPDSListSetCompareProc()*によって定義される比較手続きが、並べ替えの順序の決定に使用されます。比較手続きが指定されていない場合は、次のデフォルトの動作が適用されます。

- アイテムが8以下のサイズであれば、byte順にソートされる

■ アイテムが8より大きいサイズの場合は、比較手続きが定義されていないとソートは行われない。

尚、*XrtPDSListSetCompareProc()*に関しては、9.4.4を参照してください。

リストがソートされているかどうかを調べるには、*XrtPDSListIsSorted()*を呼び出してください。

```
int is_sorted;

is_sorted = XrtPDSListIsSorted(list);
```

この方法は、リストがソートされている時は0でない値、そうでなければ0を返します。

エレメントの移動

エレメントの位置をリスト内で変更する時は、*XrtPDSListMoveItemsBefore()*を呼び出してください。

```
Boolean       ok;
XrtPDSList    list;
int           destination;
int           source;
int           num_items;

source = 3; /* fourth element is the first to be moved */
num_items = 3; /* move three elements */
destination = 0; /* place before first element */
ok = XrtPDSListMoveItemsBefore(list, destination, source,
 num_items);
```

*XrtPDSListMoveItemsBefore()*は、次の四つのパラメータを必要とします。

■ 移動するアイテムを持つリスト
■ 移動先のエレメントのインデックス(0が一番目のエレメント、1が二番目…)
(移動されるエレメントのブロックは、このエレメントの直前に配置される)
■ 移動されるブロックの一番目のエレメントのインデックス
■ 移動されるエレメントの数

*XrtPDSListMoveItemsBefore()*は、移動が成功すると`True`、エラーが起きると`False`を返します。

9.4.7 ユーザが定義したデータ

XrtPDSListオブジェクトでは、リストに対してユーザがデータを定義することができます。データを定義するには、データのポインタを*XrtPDSListSetUserData()*に渡してください。

```
XrtPDSList        list;
XtPointer         user_data;

XrtPDSListSetUserData(list, user_data);
```

データによって使用されるメモリの解放は、ユーザ側で行ってください。

リストに対するデータを取得するには、*XrtPDSListGetUserData()*を呼び出してください。

```
user_data = XrtPDSListGetUserData(list);
```

リストにデータが定義されていなければ、この方法はNULLを返します。

9.5 手続きと方法

9.5.1 XrtPDSImageの方法

XrtPDSImageCopy()

イメージ・オブジェクトをコピーします。

```
XrtPDSImage
XrtPDSImageCopy(
  XrtPDSImage      image
  )
```

*image*はコピーされるイメージ。内部のpixmapがコピーされます。

XrtPDSImageCreate()

与えられたPixmap、Display、幅、高さを使用してXrtPDSImageを作成します。

```
XrtPDSImage
XrtPDSImageCreate(
  Pixmap       pixmap,
  Display      *display,
  int          width,
  int          height
  )
```

*pixmap*はXrtPDSImageのPixmap、*display*はイメージが作成されるDisplay、*width*と*height*は作成されるXrtPDSImageの幅と高さです。

XrtPDSImageDestroy()

イメージ・オブジェクトとそれに対応するメモリを破棄します。

```
void
XrtPDSImageDestroy(
  XrtPDSImage  image
  )
```

*image*は破棄されるイメージ。イメージの内部のpixmapが全て解放されます。

XrtPDSImageGetFilename()

イメージのファイル名を返します。このメモリは解放しないでください。

```
String
XrtPDSImageGetFilename(
  XrtPDSImage      image
  )
```

*image*は、名前が返されるファイルのイメージ。

XrtPDSImageGetWidth()

イメージの幅を返します。

```
int
XrtPDSImageGetWidth(
  XrtPDSImage      image
  )
```

*image*は、返される幅のイメージ。

XrtPDSImageGetHeight()

イメージの高さを返します。

```
int
XrtPDSImageGetHeight(
  XrtPDSImage      image
  )
```

*image*は、返される高さのイメージ。

XrtPDSImageGetColorDepth()

イメージの各カラー・コンポーネントに使用されるbit数を返します。(通常は8)

```
int
XrtPDSImageGetColorDepth(
  XrtPDSImage      image
  )
```

*image*は、色のビット数が返されるイメージ。

XrtPDSImageGetClipMask()

イメージのclip maskを返します。clip maskが指定されていなければXmUNSPECIFIED_PIXMAPを返します。

```
Pixmap
XrtPDSImageGetClipMask(
  XrtPDSImage      image
  )
```

*image*はclip maskが返されるイメージ。

XrtPDSImageGetImage()

イメージのpixmapを返します。指定されていなければ、XmUNSPECIFIED_PIXMAPを返します。

```
Pixmap
XrtPDSImageGetImage(
  XrtPDSImage      image
  )
```

*image*は、pixmapが返されるイメージ。

XrtPDSImageIsSet()

イメージ・オブジェクトの中のpixmapが有効であればTrueを返します。

```
Boolean
XrtPDSImageIsSet(
  XrtPDSImage      image
  )
```

*image*は、pixmapが有効であればTrueを返すイメージ。

XrtPDSImageIsClipMaskSet()

イメージ・オブジェクトの中のclip maskのpixmapが有効であればTrueを返します。

```
Boolean
XrtPDSImageIsClipMaskSet(
  XrtPDSImage      image
  )
```

*image*は、clip maskのpixmapが有効であればTrueを返すイメージ。

XrtPDSImageIsImage()

イメージがXrtPDSImageであればTrueを返します。

```
Boolean
XrtPDSImageIsImage(
  XrtPDSImage      image
  )
```

*image*は、XrtPDSImageであればTrueを返すイメージです。

9.5.2 XrtPDSListオブジェクトの方法

XrtPDSCmpDoubles()

二つの倍精度の浮動小数点数を比較します。一番目の数が二番目より大きければ1、同じであれば0、小さければ-1を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpDoubles(
  XtPointer    double1,
  XtPointer    double2
  )
```

*doubl1*と*double2*は比較される二つの数です。

XrtPDSCmpFloats()

二つの単精度の浮動小数点数を比較します。一番目の数が二番目より大きければ1、同じであれば0、小さければ-1を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpFloats(
  XtPointer    float1,
  XtPointer    float2
  )
```

*float1*と*float2*は比較される二つの数です。

XrtPDSCmpInts()

二つの整数を比較します。一番目の数が二番目より大きければ1、同じであれば0、小さければ-1を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpInts(
  XtPointer    int1,
  XtPointer    int2
  )
```

*int1*と*int2*は比較される二つの数です。

XrtPDSCmpStrings()

二つのストリングを比較します。一番目のストリングが二番目のストリングよりアルファベット順で後にあれば1、同じであれば0、前にあれば-1を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpStrings(
  XtPointer    string1,
  XtPointer    string2
  )
```

*string1*と*string2*は比較される二つのストリングです。

XrtPDSCmpStringsIgnoreCase()

二つのストリングを大文字・小文字関係なく比較します。一番目のストリングが二番目のストリングよりアルファベット順で後にあれば1、同じであれば0、前にあれば-1を返します。この手続きは、*XrtPDSListSetCompareProc()*へ渡す場合に適しています。

```
int
XrtPDSCmpStringsIgnoreCase(
  XtPointer    string1,
  XtPointer    string2
  )
```

*string1*と*string2*は比較される二つのストリングです。

XRTPDSListAdd()

(ソートされた)リストで該当する位置にアイテムを追加します。ソートの順序は、*XrtPDSListGetCompareProc()*によって返される手続きによって定義されます。この方法は、新しいノードの位置のインデックス、失敗時には`XRTPDS_LIST_INVALID_POSITION`を返します。

```
int
XrtPDSListAdd(
  XrtPDSList   list,
  XtPointer    new
  )
```

*list*はリスト・オブジェクト。*new*は新しいアイテムのポインタで、このサイズは*XrtPDSListGetItemSize()*によって返される値のサイズと同じです。

XrtPDSListAppend()

リストの最後にアイテムを追加します。この方法は、新しいノードの位置のインデックス、失敗時にはXRTPDS_LIST_INVALID_POSITIONを返します。

```
int
XrtPDSListAppend(
  XrtPDSList   list,
  XtPointer    new
  )
```

*list*はリスト・オブジェクト。*new*は新しいアイテムのポインタで、このサイズは、*XrtPDSListGetItemSize()*によって返される値のサイズと同じです。

XrtPDSListCreate()

0の長さのリストを作成し、それを返します。

```
XrtPDSList
XrtPDSListCreate(
  size_t   item_size
  )
```

*item_size*は、リスト上のアイテムの想定されるサイズです。

XrtPDSListDelete()

指定された位置でのアイテムを削除します。削除されたアイテムの位置を返します。

```
int
XrtPDSListDelete(
  XrtPDSList   list,
  int          position
  )
```

*list*はリスト・オブジェクト。*position*は削除されるアイテムの位置です。

破棄の手続きが(*XrtPDSSetDestroyProc()*によって)登録されている場合は、アイテムが削除される時にその手続きが呼び出されます。(破棄の手続きを呼ばずにアイテムを削除するには、*XrtPDSListRemove()*を使用してください。)

XrtPDSListDeleteALL()

リストからすべてのアイテムを削除します。

```
void
XrtPDSListDeleteAll(
  XrtPDSList   list,
  )
```

*list*は削除されるアイテムを持つリスト・オブジェクトです。

破棄の手続きが(*XrtPDSSetDestroyProc()*によって)登録されている場合は、アイテムが削除される度にその手続きが呼び出されます。(破棄の手続きを呼ばずにアイテムを削除するには、*XrtPDSListRemoveAll()*を使用してください。)

XrtPDSListDestroy()

指定されたリストを破棄し、割り当てられていたメモリを解放します。

```
void
XrtPDSListDestroy(
  XrtPDSList   list
  )
```

*list*は破棄されるリスト・オブジェクトです。

XrtPDSListDestroyObject()

指定されたリストのオブジェクトを破棄します。

```
void
XrtPDSListDestroyObject(
  XrtPDSList   list,
  XrtPDSList   *object
  )
```

*list*は破棄されるオブジェクトを持つリスト・オブジェクト、*object*は破棄されるオブジェクトです。

XrtPDSListDestroyXtPoiter()

指定されたリストのオブジェクトではないアイテムを破棄します。

```
void
XrtPDSListDestroyXtPointer(
  XrtPDSList   list,
  XtPointer    *item
  )
```

*list*は破棄されるアイテムを持つリスト・オブジェクト、*item*は破棄されるアイテムです。

XrtPDSListDuplicate()

リストのコピーを作成し、そのコピーを返します。新しいリスト・アイテムの破棄の手続きはNULLにセットされます。(*XrtPDSListSetDestroyProc()*でセットすることもできます。)

```
XrtPDSList
XrtPDSListDuplicate(
  XrtPDSList   list
  )
```

*list*はコピーされるリスト・オブジェクトです。

XrtPDSListFind()

(*XrtPDSListGetCompareProc()*によって返された手続きを使用して)指定されたアイテムに適合するリスト・アイテムを探し出します。この方法は、見つかったアイテムのインデックスを返すか、何も見つからなければXRTPDS_LIST_ITEM_NOT_FOUNDを返します。

```
int
XrtPDSListFind(
  XrtPDSList   list,
  XtPointer    item
  )
```

*list*は検索されるリスト・オブジェクト、*item*は適合するアイテムのポインタです。リストがソートされている場合(*XrtPDSIsSorted()*が0でない値を返す場合)、バイナリによる検索が行われます。

リストに重複するアイテムがある場合は、最初に来るアイテムが返されます。

XrtPDSListGetCompareProc()

オブジェクトが検索とソートに使用する手続きを取得します。

```
XrtPDSListItemCompareProc
XrtPDSListGetCompareProc(
  XrtPDSList   list
  )
```

*list*は、比較手続きが返されるリスト・オブジェクトです。

XrtPDSListGetItem()

指定されたインデックスのアイテムのポインタを返します。インデックスが無効な場合はNULLを返します。返されたアイテムはdereferenceしてください。

```
XtPointer
XrtPDSListGetItem(
  XrtPDSList   list,
  int          index
  )
```

*list*はリスト・オブジェクト、*index*は指定するアイテムのインデックス(0は一番目のエレメント、1は 二番目…と続きます)。

XrtPDSListGetItemCount()

リスト上のアイテムの数を取得します。

```
int
XrtPDSListGetItemCount(
  XrtPDSList   list
  )
```

*list*は、アイテム数がカウントされるリスト・オブジェクトです。

XrtPDSListGetItems()

アイテム配列のポインタを返します。リストが空であればNULLを返します。

```
XtPointer
XrtPDSListGetItems(
XrtPDSList list
)
```

*list*はリスト・オブジェクトです。

XrtPDSListGetItemSize()

各アイテムのサイズを返します。

```
int
XrtPDSListGetItemSize(
  XrtPDSList   list
  )
```

*list*は、サイズが返されるアイテムを持つリストです。

XrtPDSListGetUserData()

XrtPDSListオブジェクトに対するユーザ・データを取得し、それを返します。

```
XtPointer
XrtPDSListGetUserData(
  XrtPDSList   list
  )
```

*list*は、取得されるデータを持つリストです。

XrtPDSListInsertAfter()

指定された位置の後に新しいアイテムを挿入します。新しいノードのインデックスまたは、失敗時にはXRTPDS_LIST_INVALID_POSITIONを返します。

```
int
XrtPDSListInsertAfter(
  XrtPDSList   list,
  int          index,
  XtPointer    new
  )
```

*list*はリスト・オブジェクト、*index*は新しいアイテムが追加される位置の前のインデックス、*new*は新しいアイテムのポインタです。*index*がリスト内のエレメントの数より大きい場合は、*new*はリストの最後尾に追加されます。(*new*は、*XrtPDSListGetItemSize()*で返される値と同じサイズにしてください。)

XrtPDSListInsertBefore()

指定された位置の前に新しいアイテムを挿入します。新しいノードのインデックスまたは、失敗時にはXRTPDS_LIST_INVALID_POSITIONを返します。

```
int
XrtPDSListInsertBefore(
  XrtPDSList   list,
  int          index,
  XtPointer    new
  )
```

*list*はリスト・オブジェクト、*index*は新しいアイテムが追加される位置の後のインデックス、*new*は新しいアイテムのポインタです。*index*がリスト内のエレメントの数より大きい場合は、*new*はリストの最後に追加されます。(*new*は*XrtPDSListGetItemSize()*によって返される値と同じサイズにしてください。)

XrtPDSListIsList()

渡されるXrtPDSListがXrtPDSListオブジェクトであればTrueを返します。

```
Boolean
XrtPDSListIsList(
  XrtPDSList   object
  )
```

*object*はチェックされるオブジェクトです。

XrtPDSListIsSorted()

リストが現在ソートされている状態であるかを示します。ソートされていれば0ではない値を返します。

```
int
XrtPDSListIsSorted(
  XrtPDSList   list
  )
```

*list*はチェックされるリスト・オブジェクトです。

XrtPDSListLookupItem()

指定されたリストでアイテムを探し出します。このルーチンは、*XrtPDSListGetCompareProc()*によって返された手続きは使用せず、メモリを厳密に調べていきます。この方法は、適合されたアイテムの位置または、XRTPDS_LIST_ITEM_NOT_FOUNDを返します。

```
int
XrtPDSListLookupItem(
XrtPDSList    list,
XtPointer     item
)
```

*list*は検索されるリスト、*item*は探し出されるアイテムのポインタです。

XrtPDSListMoveItemsAfter()

指定された数のアイテムを移動します。これらのアイテムは、指定された位置の後に配置されます。

```
Boolean
XrtPDSListMoveItemsAfter(
  XrtPDSList   list,
  int          destination,
  int          source,
  int          num_items_to_move
)
```

*list*はリスト・オブジェクト、*destination*は移動先で、移動されるアイテムはこの後に配置されます。*source*は移動される最初のアイテムの位置、*num_items_to_move*は移動されるアイテムの数です。

XrtPDSListMoveItemsBefore()

指定された数のアイテムを移動します。これらのアイテムは、指定された位置の前に配置されます。

```
Boolean
XrtPDSListMoveItemsBefore(
  XrtPDSList   list,
  int          destination,
  int          source,
  int          num_items_to_move
  )
```

*list*はリスト・オブジェクト、*destination*は移動先で、移動されるアイテムはこの前に配置されます。*source*は移動される最初のアイテムの位置、*num_items_to_move*は移動されるアイテムの数です。

XrtPDSListRemove()

指定された位置のアイテムを除去します。除去されたアイテムの位置を返します。

```
int
XrtPDSListRemove(
  XrtPDSList   list,
  int          position
  )
```

*list*はリスト・オブジェクト、*position*は除去されるアイテムの位置です。

(*XrtPDSSetDestroyProc()*によって登録される)破棄の手続きは、アイテムが除去される時には呼び出されません。アイテムを削除し、破棄の手続きを呼び出すには、*XrtPDSListDelete()*を使用してください。

XrtPDSListRemoveAll()

リストから全てのアイテムを除去します。

```
void
XrtPDSListRemoveAll(
  XrtPDSList   list
  )
```

*list*は、除去されるアイテムを持つリスト・オブジェクトです。

(*XrtPDSSetDestroyProc()*によって登録される)破棄の手続きは、アイテムが除去される時には呼び出されません。各アイテムを削除し、破棄の手続きを呼び出すには、*XrtPDSListDeleteAll()*を使用してください。

XrtPDSListSearchForward()

指定された位置から前方向にリスト上を検索し、条件と適合するアイテムを探し出します。*XrtPDSListGetCompareProc()*によって返される手続きが使用されます。適合したアイテムの位置または、XRTPDS_LIST_ITEM_NOT_FOUNDを返します。

```
int
XrtPDSListSearchForward(
  XrtPDSList   list,
  int          start_position,
  XtPointer    item
  )
```

*list*は検索されるリスト、*start_position*は比較される最初のアイテムの位置、*item*は適合されるアイテムのポインタです。

XrtPDSListSearchReverse()

指定された位置から後方向にリスト上を検索し、条件と適合するアイテムを探し出します。*XrtPDSListGetCompareProc()*によって返される手続きが使用されます。適合したアイテムの位置または、XRTPDS_LIST_ITEM_NOT_FOUNDを返します。

```
int
XrtPDSListSearchReverse(
  XrtPDSList   list,
  int          start_position,
  XtPointer    item
  )
```

*list*は検索されるリスト、*start_position*は比較される最初のアイテムの位置、*item*は適合されるアイテムのポインタです。

XrtPDSListSetCompareProc()

リスト・オブジェクトがソートに使用する手続きを指定します。

```
void
XrtPDSListSetCompareProc(
  XrtPDSList                    list,
  XrtPDSListItemCompareProccompare_proc
  )
```

*list*は比較手続きがセットされるリスト・オブジェクト、*compare_proc*は呼び出される手続きです。

比較手続きは、次のようにして定義されます。

```
int (*XrtPDSListItemCompareProc) (
  XtPointer    item1,
  XtPointer    item2
  );
```

*item1*と*item2*は比較される二つのアイテムです。比較手続きは、下記のうちどれかを返します。

- *item1*が*item2*より大きい場合は正の整数
- 二つのアイテムが同じであれば0
- *item1*が*item2*より小さければ負の整数

XrtPDSListSetDestroyProc()

リスト・オブジェクトがアイテムを破棄する時に使用する手続きを指定します。

```
void
XrtPDSListSetDestroyProc(
  XrtPDSList                    list,
  XrtPDSListItemDestroyProcdestroy_proc
  )
```

*list*はアイテム破棄の手続きがセットされるリスト・オブジェクト。*destroy_proc*は呼び出される手続きです。

破棄の手続きは、次のようにして定義されます。

```
void (*XrtPDSListItemDestroyProc) (
 XrtPDSList object,
 XtPointer        item,
 XtPointer        user_data
 );
```

*object*は破棄されるアイテムを持つリスト・オブジェクト、*item*は破棄されるアイテムです。*user_data*は(*XrtPDSListSetUserData()*でセットされる)データです。

XrtPDSListSetItem()

アイテムをセットします。指定されたインデックスが無効な場合は`False`を返します。

```
Boolean
XrtPDSListSetItem(
 XrtPDSList   list,
 int          index,
 XtPointer    item
 )
```

*list*はリスト・オブジェクト、*index*はセットされるアイテムの位置、*item*はセットされるアイテムのポインタです。

XrtPDSListSetUserData()

XrtPDSListオブジェクトに対するユーザ・データをセットします。

```
void
XrtPDSListSetUserData(
 XrtPDSList   list,
 XtPointer    user_data
 )
```

*list*はリスト・オブジェクト、*user_data*はユーザ・データです。

XrtPDSListSort()

アイテムをソートします。リストが既にソートされている場合は、何もしません。

```
void
XrtPDSListSort(
 XrtPDSList   list
 )
```

*list*はソートされるリストです。

Part II

Reference Appendices

A

リソースの一覧

ウィジェットの概要

XRT/3dのリソース

継承されるリソース

XRT/3dの全てのリソースのアルファベット順の一覧です。リソース名の後は、データ型、デフォルト値、そしてリソースと共に使用される手続きです。**C**は、そのリソースがcreate(例: *XtCreateWidget()*)で定義でき、**S**はset(例: *XtSetValues()*)でき、**G**はget(例: *XtGetValues()*)で使用できることを意味します。

A.1 ウィジェットの概要

インクルードファイル:	*<Xm/Xrt3d.h>*
クラス名:	XtXrt3d
クラス階層:	Core → XmPrimitive → XtXrt3d
クラス・ポインタ:	xtXrt3dWidgetClass
インスタンスの生成:	`widget = XtCreateWidget(name,xtXrt3dWidgetClass ...)`
関数/マクロ:	*XtCreateXrt3d()*、*XtIsXrt3d()*

A.2 XRT/3dのリソース

XtNxrt3dAxisStrokeFont Xrt3dStrokeFont XRT3D_SF_ROMAN_SIMPLEX CSG

軸の注釈に使用されるフォントを指定します。Appendix Eにあるstroke fontにセットすることができます。

XtNxrt3dAxisStrokeSize Dimension 80 CSG

軸の注釈のフォントのサイズを指定します。サイズは、キューブの大きさに対して千単位で測定され、範囲は0から1000です。キューブの解説については、2.2をご覧ください。

XtNxrt3dAxisTitleStrokeFont Xrt3dStrokeFont XRT3D_SF_ROMAN_SIMPLEX CSG

軸のタイトルに使用されるフォントを指定します。Appendix Eにあるstroke fontにセットすることができます。

XtNxrt3dAxisTitleStrokeSize Dimension 80 CSG

軸のタイトルのフォントのサイズを指定します。サイズは、キューブの大きさに対して千単位で測定され、範囲は0から1000です。キューブの解説については、2.2をご覧ください。

XtNxrt3dBackgroundColor char* "White" CSG

ウィンドウの背景色を指定します。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrt3dContourStyles Xrt3dContourStyle** dynamic CSG

サーフェスと棒グラフで、`Xrt3dContourStyle`構造体は、ゾーンの塗りつぶし色と、等高線の色と幅、スタイルを定義します。

`XtNxrt3dContourStyles`を用いて、現在定義されている等高線スタイルを調べたり、新しいスタイルを定義することができます。新たな等高線スタイルをセットする場合は、等高線スタイル構造体のポインタの`NULL`終端配列を作成し、`XtNxrt3dContourStyles`をこの配列のポインタにセットします。*XtGetValues()*で使用されると、`Xrt3dContourStyle`構造体の配列のポインタが返されます。

実際に使用されるスタイルの数(*nlevels*)は、`XtNxrt3dDistnMethod`と`XtNxrt3dNumDistnLevels`に依存します。リニア分布が使用される場合は、*nlevels*は`XtNxrt3dNumDistnLevels`の値です。分布表(distribution table)が使用される場合は、*nlevels*はその分布表のエントリー数です。

等高線スタイルが必要数以上提供されていると、XRT/3dはそれらのスタイルから必要なスタイルを平均的に選択します。*nstyles*等高線スタイルが提供されている場合は、各レベル(*i*)で使用されるスタイルのインデックスは、次のように計算されます。[1]

```
⌊ i x (nstyles - 1) / nlevels ⌋
```

スタイルが足りない時は、提供されているスタイルから再利用します。

等高線スタイルが`XtNxrt3dContourStyles`で指定されていなければ、デフォルトのスタイルが使用されます。デフォルトのスタイルは、ディスプレイの種類によって決定されます。モノクロのサーバでは、greyscaleが使用され、黒の濃淡でデータを表示します。デフォルトのスタイルには100のエントリーがあります。`XtNxrt3dContourStyles`をセットした後で、デフォルトのスタイルに戻すには、*Xrt3dResetContourStyle()*を呼び出してください。

Xrt3dDupContourStyles() と *Xrt3dFreeContourStyles()* の手続き、*Xrt3dGetNthContourStyle()* と *Xrt3dSetNthContourStyle()*の方法は、等高線スタイル構造体の操作に使用できます。

XtNxrt3dDataAreaBackgroundColor char* NULL CSG

ユニット・キューブの側面の背景色を指定します。`NULL`であれば、無色です。ストリングは、色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、黒以外の色を指定すると白になります。

XtNxrt3dDataStyles Xrt3dDataStyles ** dynamic CSG

`Xrt3dDataStyle`構造体は、スキャッター・プロットでseriesに対する線とシンボルのスタイルを定義します。XRT/3dは、スキャッタ・プロットが参照するデフォルトのデータ・スタイルのリストを持ちます。このリソースをセットすると副作用で`XtNxrt3dDataStyleUseDefault`が`False`にセットされます。

XtNxrt3dDataStylesUseDefaultBoolean False CSG

`True`の場合は、デフォルトのデータ・スタイルが使用されます。`XtNxrt3dDataStyles`がセットされた場合は、このリソースは自動的に`False`にセットされます。

XtNxrt3dDebug Boolean False CSG

`True`の場合は、*stderr*にアプリケーションのデバッグのメッセージが出力されます。

XtNxrt3dDistnMethod Xrt3dDistnMethod(enum) XRT3D_DISTN_LINEAR CSG

サーフェスと棒グラフで、分布(distribution)法の種類を指定します。`XRT3D_DISTN_LINEAR`または`XRT3D_DISTN_FROM_TABLE`のどちらかです。(`XtNxrt3dDistnTable`を用いて)分布表が提供されている場合にのみ、`XRT3D_DISTN_FROM_TABLE`に設定できます。

[1] ⌊ x ⌋ はfloor(x)を意味し、これはx以下の最大整数を指します。

XtNxrt3dDistnTable　Xrt3dDistnTable*　NULL　CSG

任意の分布表(distribution table)を指定する時に使用するリソースです。Xrt3dDistnTable構造体の*nentries*は、表のエントリー数を指定します。*entry*は分布を指定するdoubleの配列のポインタです。

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

XtNxrt3dDistnMethodがXRT3D_DISTN_FROM_TABLEにセットされ、分布表が提供されると、この分布表が分布の定義に使用されます。一番目のエントリー以下のサーフェス・データ値は、一番目の等高線スタイルを使用して一番目のゾーンに表示されます。n-1thエントリーより大きく、nthエントリー以下のサーフェス・データ値は、nthゾーンに表示されます。最後のエントリーより大きいサーフェス・データ値は、最後の等高線スタイルを使用して*nentries*+1thゾーンに表示されます。

グラフのZ範囲外で、XtNxrt3dLegendLabelFilterでフィルタリングされたエントリーは、凡例に表示されません。XRT/3dは、エントリーを使用する前にソートしますので、エントリーを指定する時は、昇順に指定する必要はありません。また、二重のエントリーは、片方のみが使用されます。

XtNxrt3dDoubleBuffer　Boolean　True　CSG

Trueの場合、グラフのアップデートはオフスクリーンで実行されてから、画面にコピーされます。この機能は画面のちらつきを減少させます。Falseに設定すると、グラフのアップデートは直接表示され、その結果、Xサーバで使用されるメモリは小さくなります。

XtNxrt3dDrawContours　Boolean　False　CSG

サーフェスと棒グラフでは、Trueの場合、XRT/3dは分布(distribution)の各エントリーを識別しながら等高線を描きます。線の色、幅、パターンは、XtNxrt3DContourStylesの値により指定されます。分布はXtNxrt3dDistnMethod、XtNxrt3dDistnTable、XtNxrt3dNumDistnLevelsによって決定されます。4D棒グラフの場合は、このリソースは無視されます。

DrawContoursがTrueで、DrawZonesがFalseの場合、XRT/3dは、各等高線の凡例のエントリーを自動的に生成します。このリソースは、DrawMesh、DrawShaded、DrawZonesと共に使用され、15の基本的なグラフのタイプを定義します。詳細は、3.2をご覧ください。

XtNxrt3dDrawDropLines　Boolean　False　CSG

Trueの場合、グラフがXRT3D_TYPE_SCATTERであればドロップ・ラインが描画されます。

XtNxrt3dDrawHiddenLines　Boolean　False　CSG

サーフェスと棒グラフでは、ユーザの視点から隠れているメッシュ、等高線、軸を表示するかどうかを指定します。Falseの場合はリアルなイメージが得られますが、Trueの方がパフォーマンスで良い結果が得られます。

このリソースは、DrawMeshとDrawContoursの両方がFalseの時や、DrawShadedまたはDrawZonesのどちらかがTrueの場合は、無視されます。このリソースは、スキャッター・プロットには関係ありません。

XtNxrt3dDrawMesh　Boolean　True　CSG

Trueで、XtNxrt3dTypeがXRT3D_TYPE_SURFACEであると、XRT/3dは、データを3Dメッシュにグラフ化します。XまたはY方向のメッシュ線は、XtNxrt3d[XY]MeshShowを用いて非表示にすることができます。隠れたメッシュ線の表示は、XtNxrt3dDrawHiddenLinesでコントロールされます。メッシュの上部と下部の色は、XtNxrt3dMeshTopColorとXtNxrt3dMeshBottomColorで指定されます。

Trueで、XtNxrt3dTypeがXRT3D_TYPE_BARであると、XRT/3dは、各棒のアウトラインを描きます。XtNxrt3dZOriginより大きい値の棒には、XtNxrt3dMeshTopColorが使用され、XtNxrt3dZOriginより小さい値の棒には、XtNxrt3dMeshBottomColorが使用されます。

隠線の処理は、XtNxrt3dDrawHiddenLinesによってコントロールされます。

このリソースは、DrawShaded、DrawContours、DrawZonesと共に使用され、15の基本的なグラフのタイプを定義します。このリソースは、スキャッター・プロットには関係ありません。

XtNxrt3dDrawShaded　Boolean　False　CSG

Trueの場合、XRT/3dはフラットな(濃淡のない)影を使用し、3Dサーフェス(XtNxrt3dTypeがXRT3D_TYPE_SURFACE)、または3D棒グラフ(XtNxrt3dTypeがXRT3D_TYPE_BAR)を表示します。3Dサーフェスの上部と下部の色、または棒グラフのXtNxrt3dZOriginより大きい値の棒と小さい値の棒の色は、それぞれXtNxrt3dSurfaceTopColorとXtNxrt3dSurfaceBottomColorで指定されます。

このリソースは、DrawMesh、DrawContours、DrawZonesと共に使用され、15の基本的なグラフのタイプを定義します。このリソースは、スキャッター・プロットには関係ありません。

XtNxrt3dDrawZones　Boolean　False　CSG

Trueの場合、XRT/3dは分布(distribution)の各エントリーを識別しながら、ゾーン(等高線の間の領域)を塗りつぶします。ゾーンの色は、XtNxrt3dContourStylesによって決定されます。分布は、XtNxrt3dDistnMethod、XtNxrt3dDistnTable、XtNxrt3dNumDistnLevelsによって決定されます。通常、XtNxrt3dSurfaceDataがゾーンの決定に使用されますが、XtNxrt3dZoneDataが定義されていると、それが使用され、その結果4Dが表示されます。4Dのサーフェスと棒グラフの詳細は8.1と8.2をご覧ください。

DrawZonesがTrueの場合、XRT/3dは各ゾーン色の凡例のエントリーを自動的に生成します。

このリソースは、DrawMesh、DrawShaded、DrawContoursと共に使用され、15の基本的なグラフのタイプを定義します。このリソースは、スキャッター・プロットには関係ありません。

XtNxrt3dExposeCallback　XtCallbackList　NULL　CS

XRT/3dがエクスポーズ・イベントを受け取り、処理を終えた時に呼ばれるコールバックのリストを指定します。これらのコールバックは、XRT/3dがスクリーンのイメージを変更、再描画する度に呼び出されます。

*XtAddCallback()*を用いて、このリストにコールバックを加えてください。コールバックのフォーマットは、次の通りです。

```
void
CallbackProcedure(
  Widget  graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dCallbackStruct
)
```

Xrt3dCallbackStructは次のように定義されます。

```
typedef struct {
  int     reason;
  XEvent  *event;
  Window  window;
  Region  region;
  int     change;
} Xrt3dCallbackStruct;
```

*reason*はXmCR_EXPOSEです。ウィジェットが、エクスポーズ・イベントの結果、再描画される場合、*region*は再描画が必要なウィジェットの領域です。それ以外の場合は、*region*はNULLです。

*change*は、実際にウィジェットのどの領域が再描画されたかを記述します。これは、次の値から構成されます。

```
#define XRT3D_GRAPH_CHANGED 0x01
#define XRT3D_HEADER_CHANGED 0x02
#define XRT3D_FOOTER_CHANGED 0x04
#define XRT3D_LEGEND_CHANGED 0x08
#define XRT3D_DATA_CHANGED 0x19
#define XRT3D_ALL_CHANGED 0x1F /*All of the above */
```

データが変更された時は、グラフと凡例も再描画されます。無限ループを避けるために、*XtSetValues()*のコールが、エクスポーズ・コールバック内から行われた場合は、エクスポーズ・コールバックのリストは呼び出されません。

XtNxrt3dFooterAdjust　Xrt3dAdjust(enum)　XRT3D_ADJUST_CENTER　CSG

フッター領域で、テキストをどのように行揃えするかを指定します。XRT3D_ADJUST_CENTER、XRT3D_ADJUST_LEFT、XRT3D_ADJUST_RIGHTのどれかを指定します。

XtNxrt3dFooterBackgroundColor　char*　NULL　CSG

フッター領域の背景色を指定します。NULLの場合は、無色です。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrt3dFooterBorder　Xrt3dBorder(enum)　XRT3D_BORDER_NONE　CSG

フッター領域の枠線スタイルを指定します。有効な値は、XRT3D_BORDER_NONE、XRT3D_BORDER_PLAIN、XRT3D_BORDER_3D_IN、XRT3D_BORDER_3D_OUT、XRT3D_BORDER_3D_ETCHED_IN、XRT3D_BORDER_3D_ETCHED_OUT、XRT3D_BORDER_FRAME_IN、XRT3D_BORDER_FRAME_OUT、XRT3D_BORDER_BEVEL、XRT3D_BORDER_SHADOWです。

XtNxrt3dFooterBorderWidth	Dimension	2	CSG

フッター領域の枠線の幅をpixelで指定します。範囲は0から20です。

XtNxrt3dFooterFont	XmFontList	“fixed”	CSG

フッターのストリングに使用するフォントを指定します。

XtNxrt3dFooterForegroundColor	char*	NULL	CSG

フッター領域の前景色を指定します。NULLである時は、ウィンドウの前景色が使用されます。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dFooterImage	Xrt3dImage	NULL	CSG

フッター領域の背景に表示するイメージを指定します。詳細はP38を参照してください。

XtNxrt3dFooterImageLayout	Xrt3dImageLayout	XRT3D_IMAGELAYOUT_CENTERED	CSG

背景のイメージのレイアウトを指定します。有効な値はXRT3D_IMAGELAYOUT_CENTEREDかXRT3D_IMAGELAYOUT_TILEDです。

XtNxrt3dFooterImageMinimumSize	Boolean	False	CSG

フッタ領域の最小サイズが背景のイメージによって決定されるかどうか指定します。

XtNxrt3dFooterStrings	char**	NULL	CSG

フッター領域に表示されるストリングのNULL終端リストを指定します。

XtNxrt3dFooterWidth	Dimension	dynamic	G
XtNxrt3dFooterHeight	Dimension	dynamic	G

フッター領域の幅と高さ(pixel)です。

XtNxrt3dFooterX	Position	**グラフ領域の下中央に配置**	CSG
XtNxrt3dFooterY	Position	**グラフ領域の下中央に配置**	CSG

ウィンドウの左上角に対応する、フッターの左上角のx、yオフセット(pixel)を指定します。例えば、(10, 70)であれば、フッターはウィンドウの左上角から10 pixel右、70 pixel下の位置に配置されます。この設定の副作用で、対応するUseDefaultのリソースはFalseにセットされます。

XtNxrt3dFooterXUseDefault	Boolean	True	CSG
XtNxrt3dFooterYUseDefault	Boolean	True	CSG

Trueであれば、XtNxrt3dFooterXとXtNxrt3dFooterYの値は、XRT/3dによって決定されます。XtNxrt3dFooterXとXtNxrt3dFooterYの値が明示的に指定されている場合は、これらのリソースはFalseにセットされます。

XtNxrt3dForegroundColor char* "Black" CSG

ヘッダー、フッター、凡例のテキストに使用されるウィンドウ前景色を指定します。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dGraphBackgroundColor char* NULL CSG

グラフ領域の背景色を指定します。XtNxrt3dGraphBackgroundColorとXtNxrt3dGraphImageが両方ともNULLの場合は、グラフ領域は無色になります。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrt3dGraphBorder Xrt3dBorder(enum) XRT3D_BORDER_NONE CSG

グラフ領域の枠線スタイルを指定します。有効な値は、XRT3D_BORDER_NONE、XRT3D_BORDER_PLAIN、XRT3D_BORDER_3D_IN、XRT3D_BORDER_3D_OUT、XRT3D_BORDER_3D_ETCHED_IN、XRT3D_BORDER_3D_ETCHED_OUT、XRT3D_BORDER_FRAME_IN、XRT3D_BORDER_FRAME_OUT、XRT3D_BORDER_BEVEL、XRT3D_BORDER_SHADOWです。

XtNxrt3dGraphBorderWidth Dimension 2 CSG

グラフ領域の枠線の幅をpixelで指定します。範囲は0から20です。

XtNxrt3dGraphForegroundColor char* NULL CSG

グラフ領域の前景色を指定します。NULLの場合は、ウィンドウの前景色が使用されます。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dGraphImage Xrt3dImage NULL CSG

グラフ領域の背景に表示するイメージを指定します。詳細はP38を参照してください。

XtNxrt3dGraphImageLayout Xrt3dImageLayout XRT3D_IMAGELAYOUT_CENTERED CSG

背景のイメージのレイアウトを指定します。有効な値はXRT3D_IMAGELAYOUT_CENTEREDかXRT3D_IMAELAYOUT_TILEDです。

XtNxrt3dGraphWidth Dimension dynamic CSG
XtNxrt3dGraphHeight Dimension dynamic CSG

グラフ領域の幅と高さをpixelで指定します。この設定の副作用で対応するUseDefaultのリソースはFalseにセットされます。

XtNxrt3dGraphWidthUseDefault	Boolean	True	CSG
XtNxrt3dGraphHeightUseDefault	Boolean	True	CSG

Trueの場合、XtNxrt3dGraphWidthとXtNxrt3dGraphHeightの値は、XRT/3dによって決定されます。XtNxrt3dGraphWidthとXtNxrt3dGraphHeightの値が明示的に指定されていると、これらはFalseにセットされます。

XtNxrt3dGraphX	Position	**データから計算**	CSG
XtNxrt3dGraphY	Position	**データから計算**	CSG

ウィンドウの左上角に対応する、グラフ領域の左上角のx、yのオフセット(pixel)を指定します。例えば、(10, 5)の場合、グラフ領域は、ウィンドウの左上角から10 pixel右、5 pixel下の位置に配置されます。この設定の副作用で、対応するUseDefaultのリソースはFalseにセットされます。

XtNxrt3dGraphXUseDefault	Boolean	True	CSG
XtNxrt3dGraphYUseDefault	Boolean	True	CSG

Trueの場合、XtNxrt3dGraphXとXtNxrt3dGraphYの値は、XRT/3dによって決定されます。XtNxrt3dGraphXとXtNxrt3dGraphYの値が明示的に指定されていると、これらはFalseにセットされます。値が指定されていないと、これらは、Falseにできません。

XtNxrt3dHeaderAdjust	Xrt3dAdjust(enum)	XRT3D_ADJUST_CENTER	CSG

ヘッダーのテキストをどのように行揃えするかを指定します。XRT3D_ADJUST_CENTER、XRT3D_ADJUST_LEFT、XRT3D_ADJUST_RIGHTのどれかです。

XtNxrt3dHeaderBackgroundColor	char*	NULL	CSG

ヘッダー領域の背景色を指定します。XtNxrt3dHeaderBackgroundColorとXtNxrt3dHeaderImageが両方ともNULLの場合は、無色になります。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrt3dHeaderBorder	Xrt3dBorder(enum)	XRT3D_BORDER_NONE	CSG

ヘッダー領域の枠線スタイルを指定します。有効な値は、XRT3D_BORDER_NONE、XRT3D_BORDER_PLAIN、XRT3D_BORDER_3D_IN、XRT3D_BORDER_3D_OUT、XRT3D_BORDER_3D_ETCHED_IN、XRT3D_BORDER_3D_ETCHED_OUT、XRT3D_BORDER_FRAME_IN、XRT3D_BORDER_FRAME_OUT、XRT3D_BORDER_BEVEL、XRT3D_BORDER_SHADOWです。

XtNxrt3dHeaderBorderWidth	Dimension	2	CSG

ヘッダー領域の枠線幅をpixelで指定します。範囲は0から20です。

XtNxrt3dHeaderFont	XmFontList	“fixed”	CSG

ヘッダーのストリングに使用されるフォントを指定します。

XtNxrt3dHeaderForegroundColor char* NULL CSG

ヘッダー領域の前景色を指定します。NULLの場合、ウィンドウの前景色が使用されます。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dHeaderImage Xrt3dImage NULL CSG

ヘッダー領域の背景に表示するイメージを指定します。詳細はP38を参照してください。

XtNxrt3dHeaderImageLayout Xrt3dImageLayout XRT3D_IMAGELAYOUT_CENTERED CSG

背景のイメージのレイアウトを指定します。有効な値は、XRT3D_IMAGELAYOUT_CENTEREDかXRT3D_IMAGELAYOUT_TILEDです。

XtNxrt3dHeaderImageMinimumSize Boolean False CSG

ヘッダー領域の最小サイズを背景のイメージによって決定するかどうかを指定します。

XtNxrt3dHeaderStrings char** NULL CSG

ヘッダー領域に使用されるストリングのNULL終端リストを指定します。

XtNxrt3dHeaderWidth Dimension dynamic G
XtNxrt3dHeaderHeight Dimension dynamic G

ヘッダー領域の高さと幅(pixel)です。

XtNxrt3dHeaderX Position **グラフ領域の上中央に配置** CSG
XtNxrt3dHeaderY Position **グラフ領域の上中央に配置** CSG

ウィンドウの左上角に対応する、ヘッダーの左上角のx、yのオフセット(pixel)を指定します。例えば、(10, 20)の場合、ヘッダーは、ウィンドウの左上角の10 pixel右、20 pixel下の位置に配置されます。この設定の副作用で、対応するUseDefaultのリソースはFalseにセットされます。

XtNxrt3dHeaderXUseDefault Boolean True CSG
XtNxrt3dHeaderYUseDefault Boolean True CSG

Trueの場合、XtNxrt3dHeaderXとXtNxrt3dHeaderYの値は、XRT/3dによって決定されます。XtNxrt3dHeaderXとXtNxrt3dHeaderYの値が明示的に指定されていると、これらのリソースは、Falseにセットされます。

XtNxrt3dImage Xrt3dImage NULL CSG

ウィジェットの背景に表示するイメージを指定します。詳細はP38を参照してください。

XtNxrt3dImageLayout Xrt3dImageLayout XRT3D_IMAGELAYOUT_CENTERED CSG

背景のイメージのレイアウトを指定します。有効な値はXRT3D_IMAGELAYOUT_CENTEREDかXRT3D_IMAGELAYOUT_TILEDです。

XtNxrt3dLegendAnchor Xrt3dAnchor(enum) XRT3D_ANCHOR_EAST CSG

凡例の位置をアンカーで指定します。有効な値は、XRT3D_ANCHOR_NORTH, XRT3D_ANCHOR_SOUTH, XRT3D_ANCHOR_EAST, XRT3D_ANCHOR_WEST, XRT3D_ANCHOR_NORTHWEST, XRT3D_ANCHOR_NORTHEAST, XRT3D_ANCHOR_SOUTWEST, XRT3D_ANCHOR_SOUTHEASTです。

XtNxrt3dLegendBackgroundColor char* NULL CSG

凡例の背景色を指定します。XtNxrt3dLegendBackgroundColorとXtNxrt3dLegendImageが両方ともNULLの場合は、無色です。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrt3dLegendBorder Xrt3dBorder(enum) XRT3D_BORDER_NONE CSG

凡例領域の枠線のスタイルを指定します。有効な値は、XRT3D_BORDER_NONE、XRT3D_BORDER_PLAIN、XRT3D_BORDER_3D_IN、XRT3D_BORDER_3D_OUT、XRT3D_BORDER_3D_ETCHED_IN、XRT3D_BORDER_3D_ETCHED_OUT、XRT3D_BORDER_FRAME_IN、XRT3D_BORDER_FRAME_OUT、XRT3D_BORDER_BEVEL、XRT3D_BORDER_SHADOWです。

XtNxrt3dLegendBorderWidth Dimension 2 CSG

凡例領域の枠線の幅をpixelで指定します。範囲は0から20です。

XtNxrt3dLegendDistnRange Xrt3dDistnRange XRT3D_DISTN_RANGE_DATA CSG

ユーザによって指定された分布表が提供されている場合、このリソースは分布の範囲をデータによって指定されるか(デフォルト)、分布表で設定されている通りにするかを指定します。

XtNxrt3dLegendFont XmFontList “fixed” CSG

凡例に使用されるフォントを指定します。

XtNxrt3dLegendForegroundColor char* NULL CSG

凡例領域の前景色を指定します。NULLの場合、ウィンドウの前景色が使用されます。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dLegendImage Xrt3dImage NULL CSG

凡例の背景に表示するイメージを指定します。詳細はP38を参照してください。

XtNxrt3dLegendImageLayout Xrt3dImageLayout XRT3D_IMAGELAYOUT_CENTERED CSG

背景のイメージのレイアウトを指定します。有効な値はXRT3D_IMAGELAYOUT_CENTEREDかXRT3D_IMAGELAYOUT_TILEDです。

XtNxrt3dLegendLabelFilter int 1 CSG

このリソースが*n*にセットされると*n*番目ごとの項目が表示されます。

XtNxrt3dLegendLabelFunc XtFunction NULL CSG

凡例のラベルを任意に提供する場合に使用されます。NULLでなければ、XRT/3dは、各レベル(0から開始)のラベルを取得するために、提供された関数を呼び出します。関数のフォーマットは次のようになります。

```
char *
MyLegendFunction(
  Widget  graph;
  int     level;
  double  zmin, zmax;
  char *  label;
  )
```

関数が提供されると、XRT/3dはそれを呼び出し、*widget*ハンドル、ラベルを必要とする*level*(0-indexed)、レベルを定義する*zmin*と*zmax*、*label*(通常デフォルトで計算される)をパラメータとして渡します。この関数は、(そのレベルの)凡例ラベルに使用されるストリングを返すようにしてください。凡例がstep(XRT3D_LEGEND_STYLE_STEPPED)され、関数がNULLを返すと、XRT/3dは、凡例のエントリーをスキップします。

ストリングには30フィールドまで含むことができ、それぞれ、*left*、*center*、*right*揃えを定義する'\l', '\c', '\r'で区切ります。(例: "\r32.0\cto\r39.6") '\'は、もう一つの'\'でエスケープしてください。

XtNxrt3dLegendOrientation Xrt3dAlign(enum) XRT3D_ALIGN_VERTICAL CSG

凡例の方向を指定します。有効な値は、XRT3D_ALIGN_HORIZONTALとXRT3D_ALIGN_VERTICALです。

XtNxrt3dLegendShow Boolean True CSG

凡例表示の有無を決定します。凡例は、等高線やゾーンが描画される時、XRT/3dによって自動的に作成されます。

XtNxrt3dLegendStrings char** NULL CSG

凡例に使用されるストリングを指定します。サーフェスと棒グラフでは、分布(distribution)レベルの数は、XtNxrt3dNumDistnLevelsまたはXtNxrt3dDistnTableの値と、XtNxrt3dLegendStyleの設定によって決定されます。スキャッター・プロットでは、各シリーズに対して一つのエントリが必要です。提供されるストリングが少ないと、対応するエントリーは無視されます。凡例は、等高線かゾーンのどちらかが描画され、XtNxrt3dLegendShowがTrueの時にのみ、生成されます。

これらの凡例ストリングは、XRT/3dが生成したものを置き換えます。但し、XtNxrt3dLegendLabelFuncが指定されている場合は、凡例ストリングは実行時にダイナミックに置き換えられます。このリソースは、4D棒グラフの特定の棒のグループを注釈する時に便利です。

XtNxrt3dLegendStyle　Xrt3dLegendStyle　XRT3D_LEGEND_STYLE_CONTINUOUS　CSG

サーフェスと棒グラフでは、凡例のスタイルを決定します。XRT3D_LEGEND_STYLE_STEPPEDかXRT3D_LEGEND_STYLE_CONTINUOUSのどちらかです。stepされた凡例では、各ゾーンの範囲がプリントされます。continuousされた凡例は、隣接したゾーンの境界にあるレベルがプリントされます。

XtNxrt3dLegendTitle　char *　NULL　CSG

このリソースによって凡例のタイトルを指定することができます。凡例のフォントを使用した一行のタイトルが凡例の上中央に表示されます。

XtNxrt3dLegendWidth　Dimension　dynamic　G
XtNxrt3dLegendHeight　Dimension　dynamic　G

凡例領域の高さと幅(pixel)です。

XtNxrt3dLegendX　Position　**アンカーの位置に依存**　CSG
XtNxrt3dLegendY　Position　**アンカーの位置に依存**　CSG

凡例領域の左上角の(X, Y)座標を指定します。この設定の副作用で、対応するUseDefaultのリソースがFalseにセットされます。

XtNxrt3dLegendXUseDefault　Boolean　True　CSG
XtNxrt3dLegendYUseDefault　Boolean　True　CSG

Trueの場合、XtNxrt3dLegendXとXtNxrt3dLegendYの値は、XRT/3dによって決定されます。XtNxrt3dLegendXとXtNxrt3dLegendYの値が明示的に指定されている時は、Falseにセットされます。

XtNxrt3dMapCallback　XtCallbackList　NULL　CSG

ユーザ・イベントのグラフ座標をアプリケーションに通知するために呼び出されるコールバックのリストを指定します。Mapコールバックは、*Xrt3dMap()*を使用する*Map()*アクション[1]によって呼び出されます。Xrt3dMapCallbackStructのポインタが各コールバックに渡されます。*XtAddCallback()*を用いてリストにコールバックを追加してください。コールバックのフォーマットは次の通りです。

```
void
CallbackProcedure(
  Widget  graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dMapCallbackStruct
)
```

[1] このアクションには、デフォルトのトランスレーションはありません。

Xrt3dMapCallbackStructは、次のように定義されます。

```
typedef struct {
  int             reason;      // Read-only
  XEvent          *event;      // Read-only
  String          *params;     // Read-only
  Cardinal        nparams;     // Read-only
  Xrt3dRegion region;      // Read-only
  Xrt3dMapResult  map;         // Read-only
} Xrt3dMapCallbackStruct;
```

*reason*は常にXRT3D_REASON_MAPです。*region*は、イベントが起こった領域です。*map*は、*Xrt3dMap()*によって計算されたグラフ座標を含んでいます。

XtNxrt3dMeshBottomColor	char*	“Black”	CSG
XtNxrt3dMeshTopColor	char*	“Black”	CSG

XtNxrt3dDrawMeshがTrueの時、3Dサーフェスのメッシュ線または棒グラフのアウトライン(輪郭)の色を指定します。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7をご覧ください。

棒グラフでは、topの色は、XtNxrt3dZOriginより大きい値の棒に使用され、bottomの色は、これより小さい値の棒に使用されます。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dModifyCallback	XtCallbackList	NULL	CSG

ユーザ・インタラクション(スケーリング、ズーム、移動、回転、リセット)の開始と終了をアプリケーションに通知するために呼ばれるコールバックのリストを指定します。Modifyコールバックは、*ModifyStart()* と *ModifyEnd()* によって呼び出されます。Xrt3dModifyCallbackStructのポインタが各コールバックに渡されます。*XtAddCallback()*を用いてリストにコールバックを追加してください。コールバックのフォーマットは、次の通りです。

```
void
CallbackProcedure(
  Widget      graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dModifyCallbackStruct
)
```

Xrt3dModifyCallbackStructは、次のように定義されます。

```
typedef struct {
  int         reason;      // Read-only
  XEvent      *event;      // Read-only
  String      *params;     // Read-only
  Cardinal    nparams;     // Read-only
  Boolean doit;
} Xrt3dModifyCallbackStruct;
```

*reason*はXRT3D_REASON_MODIFY_STARTかXRT3D_REASON_MODIFY_ENDのどちらかです。*doit*は、(*reason*がXRT_REASON_MODIFY_STARTの時)インタラクションを許可するかどうかを決定するフラグです。全てのユーザ・インタラクションを不可にするには、*reason*がXRT3D_REASON_MODIFY_STARTの時に、コールバックで*doit*をFalseにセットします。

XtNxrt3dNumDistnLevels int 10 CSG

サーフェスと棒グラフでは、DrawContoursと(または)DrawZonesがTrueの場合の、分布(distribution)レベルの数(*nlevels*)を指定します。有効な値は0から100です。0に設定すると等高線は描かれず、DrawZonesがTrueであれば、ゾーンが一つだけ表示されます。任意の分布表が提供され、XtNxrt3dDistnMethodがXRT3D_DISTN_FROM_TABLEであると、分布レベルの数は分布表のエントリー数になり、XtNxrt3dNumDistnLevelsは無視されます。

XtNxrt3dPerspectiveDepth double 2.5 CSG

透視図の表示をコントロールします。範囲は、1からXRT3D_HUGE_VALです。値が小さいほど、透視図法の効果が大きくなります。遠近感を少なく(または全くないように)するには、このリソースを高い値(100,000等)にセットしてください。

XtNxrt3dPickCallback XtCallbackList NULL CSG

ユーザ・イベントのpixel座標に一番近いデータ値をアプリケーションに通知するために呼ばれるコールバックのリストを指定します。Pickコールバックは、*Xrt3dPick()*を使用する*Pick()*アクション[1] によって呼び出されます。Xrt3dPickCallbackStructのポインタが各コールバックに渡されます。*XtAddCallback()*を用いてこのリストにコールバックを追加してください。コールバックのフォーマットは、次の通りです。

```
void
CallbackProcedure(
  Widget  graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dPickCallbackStruct
)
```

Xrt3dPickCallbackStructは、次のように定義されます。

```
typedef struct {
  int         reason;      // Read-only
  XEvent      *event;      // Read-only
  String      *params;     // Read-only
  Cardinal        nparams;     // Read-only
  Xrt3dRegion     region;      // Read-only
  Xrt3dPickResult pick;        // Read-only
} Xrt3dPickCallbackStruct;
```

*reason*は常にXRT3D_REASON_PICKです。*region*は、イベントが起きた領域です。*pick*は、*Xrt3dPick()*によって計算されたグリッド座標を持ちます。

XtNxrt3dPreviewMethod Xrt3dPreviewMethod XRT3D_PREVIEW_CUBE CSG

インタラクティブな回転をどのように表示するかを指定します。XRT3D_PREVIEW_CUBEは、ユニット・キューブのアウトライン(輪郭)だけを表示し、XRT3D_PREVIEW_FULLは、回転中でもグラフ全体を表示します(小さなグラフや棒グラフに適しています)。

[1] このアクションには、デフォルトのトランスレーションはありません。

XtNxrt3dProjectZMax	int	0	CSG
XtNxrt3dProjectZMin	in	0	CSG

サーフェスと棒グラフで、等高線と(または)ゾーンの投影を*z=zmin*または*z=zmax*の面に表示するかどうかを指定します。これらのリソースは、次の定数の組み合わせで指定されます。

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

等高線やゾーンの投影は、DrawContoursとDrawZonesの設定に関係なく作成されます。これらのリソースは、2Dグラフや、XtNxrt3dTypeがXRT3D_TYPE_BARである時は無視されます。例えば、等高線とゾーンの両方の投影を表示する場合、次のように設定します。

```
XRT3D_PROJECT_CONTOURS | XRT3D_PROJECT_ZONES
```

XtNxrt3dRepaint	Boolean	True	CSG

Trueの場合、ウィジェットに加えられた変更は直ちに表示されます。変更を一括処理するには、このリソースをFalseにセットし、変更を加えた後、最後の*XtSetValues()*のコールで、このリソースをTrueにセットします。

XtNxrt3dResizeCallback	XtCallbackList	NULL	CS

XRT/3dが、リサイズ・イベントを受け取り、処理を終えた時に呼ばれるコールバックのリストを指定します。*XtAddCallback()*を用いてリストにコールバックを加えてください。コールバックのフォーマットは次の通りです。

```
void
CallbackProcedure(
  Widget      graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dCallbackStruct
)
```

Xrt3dCallbackStructは次のように定義されます。

```
typedef struct {
  int         reason;
  XEvent      *event;
  Window      window;
  Region      region;
  int         change;
} Xrt3dCallbackStruct;
```

*reason*は、XmCR_RESIZEです。リサイズのコールバックでは、イベントは常にNULLです。*region*と*change*はリサイズ・コールバックでは使用されません。

XtNxrt3dRotateCallback	XtCallbackList	NULL	CSG

ユーザがグラフを回転している時に、アプリケーションに通知するために呼び出されるコールバックのリストを指定します。Rotateコールバックは、*RotateUpdate()*アクションによって呼び出されます。Xrt3dRotateCallbackStructのポインタが各コールバックに渡されます。*XtAddCallback()*を用いてリストにコールバックを追加してください。コールバックのフォーマットは次の通りです。

```
void
CallbackProcedure(
  Widget      graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dRotateCallbackStruct
  )
```

Xrt3dRotateCallbackStructは、次のように定義されます。

```
typedef struct {
  int         reason;      // Read-only
  XEvent      *event;      // Read-only
  String      *params;     // Read-only
  Cardinal    nparams;     // Read-only
  double      xrotation;
  double      yrotation;
  double      zrotation;
  Boolean doit;
} Xrt3dRotateCallbackStruct;
```

reason は常に XRT3D_REASON_ROTATE です。 *xrotation* 、 *yrotation* 、 *zrotation* は、XtNxrt3d[XYZ]Rotationの新しい値です。*doit*は、回転を許可するかどうかを決定するフラグです。(デフォルトはTrueです。) 回転を不可にするには、*doit*をFalseにセットしてください。新しい値を修正する場合は、構造体の値を変更します。

XtNxrt3dSolidSurface	Boolean	False	CSG

サーフェスと棒グラフでは、このリソースがTrueで、DrawMeshまたはDrawShadedがTrueの場合、XRT/3dは、3Dサーフェスに側面を加えます。側面は、サーフェスからZの最小値を結びます。メッシュとサーフェスの下部の色が側面に使用されます。このリソースは、棒グラフには、適用されません。

XtNxrt3dSurfaceBottomColor	char*	dim grey	CSG
XtNxrt3dSurfaceTopColor	char*	light grey	CSG

XtNxrt3dDrawShadedがTrueの時、3Dサーフェスと棒グラフの上部と下部の色を指定します。ストリングは色名か#で始まるRGB(赤、緑、青)構成の二桁の16進数です。

色の指定については、2.7をご覧ください。棒グラフでは、XtNxrt3dZOriginより大きい値の棒にtopの色が使用され、それより小さい値の棒には、bottomの色が使用されます。

XtNxrt3dSurfaceData	Xrt3dData*	NULL	CSG

3Dサーフェスと棒グラフに表示されるデータを指定します。Xrt3dData構造体は、*Xrt3dMakeGridData()*または*Xrt3dMakeDataFromFile()*を使用して割り当ててください。この構造体についての詳細は、6.2をご覧ください。

XtNxrt3dTextList	WidgetList	NULL	CSG

テキスト・オブジェクトのNULL終端リストを指定します。このリソースの値は、新しいテキスト・オブジェクトが作成されると自動的にアップデートされます。テキスト・オブジェクトのリストは、リソース・ファイルで指定できます。(*Xrt3dTextCreate()*を使用して、リストの各名前に新しいテキスト・オブジェクトを作成します。)

XtNxrt3dTransformCallback	XtCallbackList	NULL	CSG

ユーザがグラフを移動、スケーリング、ズームしている時に、アプリケーションに通知するために呼ばれるコールバックのリストを指定します。Transformコールバックは、*TranslateUpdate()* 、 *ZoomEnd()* 、 *ViewReset()* によって呼び出されます。Xrt3dTransformCallbackStructのポインタが各コールバックに渡されます。

*XtAddCallback()*を用いてリストにコールバックを追加してください。コールバックのフォーマットは、次の通りです。

```
void
CallbackProcedure(
  Widget      graph,
  XtPointer   client_data,
  XtPointer   call_data // points to an Xrt3dTransformCallbackStruct
)
```

Xrt3dTransformCallbackStructは、次のように定義されます。

```
typedef struct {
  int         reason;      // Read-only
  XEvent      *event;      // Read-only
  String      *params;     // Read-only
  Cardinal    nparams;     // Read-only
  double      scale;
  double      xtranslate;
  double      ytranslate;
  Boolean doit;
} Xrt3dTransformCallbackStruct;
```

reason は常に XRT3D_REASON_TRANSFORM です。 *scale* 、 *xtranslate* 、 *ytranslate* は、XtNxrt3dViewScale と XtNxrt3dView[XY]Translate の新しい値です。 *doit* は、transform(グラフの表示の変更)を許可するかどうかを決定するフラグです。(デフォルトは、Trueです。) transformを不許可にするには、*doit*をFalseにセットしてください。新しい値を修正する時は、構造体の値を変更します。

XtNxrt3dType　　Xrt3dType　　XRT3D_TYPE_SURFACE　　CSG

表示されるグラフの基本タイプを指定します。XRT3D_TYPE_SURFACEかXRT3D_TYPE_BARのどちらかです。サーフェスと棒グラフでは、四つの基本的なBooleanリソースを操作することにより、多くのバリエーションが作成できます。詳細は3.2をご覧ください。

XtNxrt3dViewNormalized	Boolean	True	CSG

`True`の場合、XRT/3dは、(スケーリングや移動等が適用される前の)グラフ全体のビュー(view)を計算し、グラフのどの部分も欠けないようにビューを最大限にします。このリソースが`True`の時に回転を行うとグラフの表示が断続的になりますので、`False`にセットしてください。

XtNxrt3dViewScale	double	1.0	CSG

グラフ全体の完全なビュー(view)からの、スケーリング(ズーム)の度合を指定します。有効な値は、1/30から30.0です。このリソースは、ユーザがグラフをスケーリングまたはズームするとアップデートされます。ダイナミックなスケーリングについては、7.4.1をご覧ください。

XtNxrt3dViewXTranslate	double	0.0	CSG
XtNxrt3dViewYTranslate	double	0.0	CSG

グラフ全体の完全なビュー(view)からの、移動の度合を指定します。移動は、起点が左下角にあるnormalized boxに対応します。最大単位より大きく(>±1.0)移動すると、ビューが視界からはずれます。このリソースは、ユーザがビューを移動するとアップデートされます。ダイナミックな移動については、7.4.1をご覧ください。

XtNxrt3dXAnnoMethod	Xrt3dAnnoMethod	XRT3D_ANNO_VALUES	CSG
XtNxrt3dYAnnoMethod	Xrt3dAnnoMethod	XRT3D_ANNO_VALUES	CSG
XtNxrt3dZAnnoMethod	Xrt3dAnnoMethod	XRT3D_ANNO_VALUES	CSG

軸の注釈法を指定します。有効な値がXRT3D_ANNO_VALUES、XRT3D_ANNO_DATA_LABELS、XRT3D_ANNO_VALUE_LABELSである列挙型です。
XRT3D_ANNO_VALUESの場合、ストリングはデータの範囲に基づいて自動的に生成されます。XRT3D_ANNO_DATA_LABELSの場合は、XtNxrt3d[XY]DataLabelsの値が使用されます。(Z軸には適用されません) XRT3D_ANNO_VALUE_LABELSの場合は、XtNxrt3d[XYZ]ValueLabelsの値が使用されます。データ・ラベルはスキャッター・プロットには使用されません。

XtNxrt3dXAxisShow	Boolean	True	CSG
XtNxrt3dYAxisShow	Boolean	True	CSG
XtNxrt3dZAxisShow	Boolean	True	CSG

X、Y、Z軸の表示の有無を決定します。

XtNxrt3dXAxisTitle	char*	NULL	CSG
XtNxrt3dYAxisTitle	char*	NULL	CSG
XtNxrt3dZAxisTitle	char*	NULL	CSG

X、Y、Z軸のタイトルを指定します。

XtNxrt3dXBarFormat	Xrt3dBarFormat	XRT3D_BAR_FIXED	CSG
XtNxrt3dYBarFormat	Xrt3dBarFormat	XRT3D_BAR_FIXED	CSG

棒グラフのフォーマットを指定します。XRT3D_BAR_FIXEDの場合、棒はX値とY値の中央に配置され、各棒の幅は固定されます。XRT3D_BAR_HISTOGRAMに設定されると、各棒は隣の棒までのスペースを占め、棒の幅は、グリッドの間隔になります。

XtNxrt3dXBarSpacing	double	80.0	CSG
XtNxrt3dYBarSpacing	double	80.0	CSG

棒(XまたはY方向)がグラフに占めるスペースを(可能範囲に対する割合で)指定します。有効な値は1.0から100.0です。値が100.0の場合は、棒が隣り合って並びます。(不規則的なグリッド・データでは隣り合わない時があります。)このリソースは、XtNxrt3dTypeがXRT3D_TYPE_BARにセットされ、XtNxrt3d[XY]BarFormatがXRT3D_BAR_FIXEDの場合にのみ、適用されます。

XtNxrt3dXDataLabels	char**	NULL	CSG
XtNxrt3dYDataLabels	char**	NULL	CSG

XtNxrt3d[XY]AnnoMethodがXRT3D_ANNO_DATA_LABELSに設定されている時に、このリソースはラベルを提供します。これは、ストリングのNULL終端リストで、各ラベルは、グリッドにあるデータの行や列に適用されます。データ・ラベルはスキャッター・プロットには使用されません。

XtNxrt3dXGridLines	int	0	CSG
XtNxrt3dYGridLines	int	0	CSG
XtNxrt3dZGridLines	int	0	CSG

これらのリソースは、各面のグリッド線を指定します。三つの面はそれぞれ定数で表され、その組み合わせで、グリッド線を表示する面を指定します。2Dグラフでは、0以外の値は、軸に垂直に交差するグリッド線を描きます。グリッド線は、軸のラベルがある箇所に描かれます。次の定数は、<Xm/*xrt3d_common.h*>に定義されており、これらのリソースの値を設定するときに使用します。

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

例えば、全ての面にXグリッド線を表示するには、`XtNxrt3dXGridLines`を次のようにセットします。

```
XRT3D_XY_PLANE | XRT3D_XZ_PLANE
```

グリッド線を消すには、これらのリソースを0にします。

XtNxrt3d[XYZ]GridLineStyle	Xrt3dLineStyle *	solid line	CSG

X、Y、Zグリッド線の色、幅、線パターンをコントロールします。このリソースをセットすると副作用で、対応する`XtNxrt3d[XYZ]GridLiineStyle`が`False`にセットされます。

XtNxrt3d[XYZ]GridLineStyleUseDefault	Boolean	True	CSG

`True`の場合、X、Y、Zグリッド線には、デフォルトのスタイルが使用されます。デフォルトは、グラフの前景色または、(グラフの前景色が指定されていなければ)ウィジェットの前景色を使用した、幅1の実線です。`XtNxrt3d[XYZ]GridLineStyle`を設定すると、このリソースは`False`にセットされます。

XtNxrt3dXMax	double	dynamic	CSG
XtNxrt3dYMax	double	dynamic	CSG
XtNxrt3dZMax	double	dynamic	CSG

表示されるX、Y、Zの最大値を指定します。これより大きいXデータ値とYデータ値は表示されません。データをズームするには、XとYの最大値と最小値の幅を狭くします。`XtNxrt3dZMax`は、最大Zデータ値より小さくできません。[1] この設定の副作用で、対応する`UseDefault`のリソースが`False`にセットされます。

XtNxrt3dXMaxUseDefault	Boolean	True	CSG
XtNxrt3dYMaxUseDefault	Boolean	True	CSG
XtNxrt3dZMaxUseDefault	Boolean	True	CSG

`True`の場合、`XtNxrt3d[XYZ]Max`の値は、XRT/3dによって決定されます。`XtNxrt3d[XYZ]Max`に明示的に値が指定されていると、これらのリソースは、`False`にセットされます。`XtNxrt3dZMax`の値が計算または提供されるまで、このリソースは`False`にセットできません。`XtNxrt3dZMax`の値がデータの最大Z値より小さいと、`XtNxrt3dZMaxUseDefault`は`True`にセットされます。

[1] このリソースの値がデータのZ範囲の内側に設定されると、XRT/3dは対応する`UseDefualt`を`True`にセットします。

XtNxrt3dXMeshFilter	int	1	CSG
XtNxrt3dYMeshFilter	int	1	CSG

表示する前に、メッシュをどのようにフィルタリング(数を減少)するか指定します。値*n*に対して各*n*番目のメッシュ線が描かれます。このリソースが0にセットされると、XRT/3dは、適するフィルタリングを設定します。フィルタリングは、規則的なグリッドと不規則的なグリッドの両方に適用されます。

これらのリソースは、DrawMeshが`True`の場合にのみ影響します。`XtNxrt3dType`が`XRT3D_TYPE_BAR`または`XRT3D_TYPE_SCATTER`であると、これらは無視されます。

XtNxrt3dXMeshShow	Boolean	True	CSG
XtNxrt3dYMeshShow	Boolean	True	CSG

`XtNxrt3dDrawMesh`が`True`の時、メッシュ線の表示の有無を指定します。棒グラフとスキャッター・プロットの場合は適用されません。

XtNxrt3dXMin	double	dynamic	CSG
XtNxrt3dYMin	double	dynamic	CSG
XtNxrt3dZMin	double	dynamic	CSG

表示されるX、Y、Zの最小値を指定します。これより小さいXデータ値とYデータ値は表示されません。データをズームするには、XとYの最小値と最大値の幅を狭くします。`XtNxrt3dZMin`は、最小Zデータ値より大きくすることはできません。[1] この設定の副作用で、対応する`UseDefault`のリソースが`False`にセットされます。

XtNxrt3dXMinUseDefault	Boolean	True	CSG
XtNxrt3dYMinUseDefault	Boolean	True	CSG
XtNxrt3dZMinUseDefault	Boolean	True	CSG

`True`の場合、`XtNxrt3d[XYZ]Min`の値は、XRT/3dによって決定されます。`XtNxrt3d[XYZ]Min`に明示的に値が指定されていると、これらのリソースは`False`にセットされます。`XtNxrt3dZMin`の値が計算または提供されるまで、このリソースは`False`にセットできません。`XtNxrt3dZMin`の値がデータの最小Z値より大きい場合、`XtNxrt3dZMinUseDefault`の値は、`True`にセットされます。

XtNxrt3dXRotation	double	45.0	CSG
XtNxrt3dYRotation	double	0.0	CSG
XtNxrt3dZRotation	double	45.0	CSG

ユニット・キューブの回転の角度を、時計と反対の方向に指定します。回転は、Z-Y-Xの順で適用されます。これらのリソースは、ユーザがキューブを回転した時にアップデートされます。ダイナミックな回転については、7.4.2をご覧ください。

[1] このリソースの値がデータのZ範囲の内側に設定されると、XRT/3dは対応する`UseDefualt`を`True`にセットします。

XtNxrt3dXScale	double	1.0	CSG
XtNxrt3dYScale	double	1.0	CSG
XtNxrt3dZScale	double	1.0	CSG

XtNxrt3d[XYZ]Scaleのリソースは、ユニット・キューブのスケーリングに使用されます。例えば、キューブをY方向に2倍に拡大する場合(XとZの方向に比べてYが2倍の長さの直方体)は、XtNxrt3dYScaleを2.0に指定します。

XtNxrt3dXValueLabels	Xrt3dValueLabel**	NULL	CSG
XtNxrt3dYValueLabels	Xrt3dValueLabel**	NULL	CSG
XtNxrt3dZValueLabels	Xrt3dValueLabel**	NULL	CSG

XtNxrt3d[XY]AnnoMethodがXRT3D_ANNO_VALUE_LABELSの時、軸のラベルを指定します。これらのラベルは、軸の特定の値に適用されるストリングのセットを提供します。詳細は、3.12をご覧ください。

XtNxrt3dXYColors	Xrt3dXYColor**	NULL	CSG

XtNxrt3dDrawShadedがTrueで、XtNxrt3dDrawZonesがFalseの時、棒グラフに適用される色のリストを指定します。これらの色は、棒の行や列全体、または各棒ごとに適用できます。詳細は、3.3にあるXrt3dXYColor構造体の解説を参照してください。

このリソースは、XtNxrt3dTypeがXRT3D_TYPE_SURFACEまたはXRT3D_TYPE_SCATTERの時には、適用されません。

XtNxrt3dZOrigin	double	0.0	CSG

XtNxrt3dTypeがXRT3D_TYPE_BARの時に棒の起点となるZ値を定義します。またXtNxrt3dTypeがXRT3D_TYPE_SCATTERの時は、スキャッター・プロットのドロップ・ラインがこの値から描画されます。このリソースは、XtNxrt3dTypeがXRT3D_TYPE_SURFACEの時には何もしません。

XtNxrt3dZoneData	Xrt3dData*	NULL	CSG

XtNxrt3dTypeがXRT3D_TYPE_SURFACEの時、このリソースは、4Dグラフのサーフェス・データ(XtNxrt3dSurfaceDataで指定)に重ねられるゾーン・データ(filled contour)を指定します。4Dグラフや投影(projection)が必要な時以外は、このリソースはNULLにします。4Dサーフェス・グラフについては、8.1を参照してください。

XtNxrt3dTypeがXRT3D_TYPE_BARの時、このリソースは、棒の色の決定に使用されるデータ・セットを指定し、その結果4D棒グラフが作成されます。4D棒グラフの解説は、6.2にあります。このリソースは、XtNxrt3dTypeがXRT3D_TYPE_SCATTERの時には何もしません。

XtNxrt3dZoneMethod	Xrt3dZoneMethod	XRT3D_ZONE_CONTOURS	CSG

サーフェスと棒グラフでは、DrawZoneがTrueの時に表示される、ゾーンの領域を塗りつぶす方法を指定します。XRT3D_ZONE_CONTOURSにセットされると、各等高線の間が塗りつぶされます。XRT3D_ZONE_CELLSの場合は、グリッドの各セルを(セルの平均値に基づいて)塗りつぶします。

A.3 継承されるリソース

XRT/3dは、親のクラスからリソースを継承します。次の表は、リソースとそれを定義する親のクラスの一覧です。詳細は、MotifやXtの資料を参照してください。

Resource	Inherited From
XmNdestroyCallback	Core
XmNheight	Core
XmNhightlightColor	XmPrimitive
XmNhighlightOnEnter	XmPrimitive
XmNhighlightPixmap	XmPrimitive
XmNnavigationType	XmPrimitive
XmNshadowThickness	XmPrimitive

Resource	Inherited From
XmNtranslations	Core
XmNtraversalOn	XmPrimitive
XmNuserData	XmPrimitive
XmNwidth	Core
XmNx	Core
XmNy	Core

B

テキスト・オブジェクトのリソースの一覧

オブジェクトの概要

テキスト・オブジェクトのリソース

継承されたリソース

テキスト・オブジェクトのリソースのアルファベット順の一覧です。リソース名の後は、データ型、デフォルト値そしてリソースと共に使用されるXtの手続き(**C**reate、**S**et、**G**et)です。

B.1 オブジェクトの概要

インクルードファイル:	必要ありません。XRT/3d(<*Xm/Xrt3d.h*>)に含まれています。
クラス名:	XtXrt3dText
クラス階層:	Object → XtXrt3dText
クラス・ポインタ:	xtXrt3dTextObjectClass
インスタンスの生成:	`widget = XtCreateWidget(name,xtXrt3dTextObjectClass ...)`
関数/マクロ:	*XtCreateXrt3dText()*、*XtIsXrt3dText()*

B.2 テキスト・オブジェクトのリソース

XtNxrt3dTextAdjust　Xrt3dAdjust　XRT3D_ADJUST_CENTER　CSG

テキスト・オブジェクト内で、複数行のテキストをどのように行揃えするか指定します。有効な値は、XRT3D_ADJSUT_LEFT、XRT3D_ADJUST_CENTER、XRT3D_ADJUST_RIGHTです。

XtNxrt3dTextAttachIndexX	int	0	CSG
XtNxrt3dTextAttachIndexY	int	0	CSG

テキスト・オブジェクトが貼り付けられるグリッド・インデックスを指定します。このリソースは、XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_INDEXの時に、テキスト・オブジェクトの位置決めに使用されます。スキャッター・プロットには使用されません。

XtNxrt3dTextAttachIndexPoint	int	0	CSG

テキスト・オブジェクトが貼り付けられるポイントを指定します。このリソースは、データがXRT3D_DATA__POINTの時に使用されます。

XtNxrt3dTextAttachIndexSeries	int	0	CSG

テキスト・オブジェクトが貼り付けられるシリーズを指定します。このリソースは、データがXRT3D_DATA_POINTの時に使用されます。

XtNxrt3dTextAttachMethod	Xrt3dTextAttachMethod	XRT3D_TEXT_ATTACH_INDEX	CSG

グラフにどのようにテキスト・オブジェクトを貼り付けるか指定します。有効な値は、XRT3D_TEXT_ATTACH_PIXEL、XRT3D_TEXT_ATTACH_POINT、XRT3D_TEXT_ATTACH_INDEXです。テキスト・オブジェクトの位置決めは、貼り付けの方法のリソース(XtNxrt3dTextAttachPixel[XY]、XtNxrt3dTextAttachPoint[XYZ]、XtNxrt3dTextAttachIndex[XY]、XtNxrt3dTextAttachIndexSeries、XtNxrt3dTextAttachIndexPoint)によってコントロールされます。

XtNxrt3dTextAttachPixelX	Position	0	CSG
XtNxrt3dTextAttachPixelY	Position	0	CSG

テキスト・オブジェクトが貼り付けられるpixel座標を指定します。このリソースは、XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_PICXELの時に、テキスト・オブジェクトの位置決めをコントロールします。

XtNxrt3dTextAttachPointX	double	0.0	CSG
XtNxrt3dTextAttachPointY	double	0.0	CSG
XtNxrt3dTextAttachPointZ	double	0.0	CSG

テキスト・オブジェクトが貼り付けられる3Dグラフ座標を指定します。このリソースは、XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_POINTの時に、テキスト・オブジェクトの位置決めをコントロールします。

XtNxrt3dTextBackgroundColor	char*	NULL	CSG

テキスト・オブジェクトの背景色を指定します。XtNxrt3dTextBackgroundColorとXtNxrt3dTextImageの両方がNULLであれば、テキスト・オブジェクトは無色です。ストリングは、色名か、#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7を参照してください。

モノクロのサーバでは、黒以外の色を指定しますと白に表示されます。

XtNxrt3dTextBorder	Xrt3dBorder	XRT3D_BORDER_NONE	CSG

テキスト・オブジェクトの枠線を指定します。このリソースは、テキスト・オブジェクトがpixelに貼り付けられている場合(XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_PIXEL)に使用されます。有効な値は、XRT3D_BORDER_NONE、XRT3D_BORDER_PLAIN、XRT3D_BORDER_3D_IN、XRT3D_BORDER_3D_OUT、

XRT3D_BORDER_ETCHED_IN、XRT3D_BORDER_ETCHED_OUT、XRT3D_BORDER_FRAME_IN、XRT3D_BORDER_FRAME_OUT、XRT3D_BORDER_BEVEL、XRT3D_BORDER_SHADOW、です。

ポイントまたはインデックスに貼り付けられたテキスト・オブジェクトは、オブジェクトの周囲に1 pixel幅の線を持ち、テキスト・オブジェクトの背景色がNULLでない時に表示されます。

XtNxrt3dTextBorderWidth Dimension 2 CSG

XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_PIXELの時に、テキスト・オブジェクトの枠線の幅を指定します。範囲は、0から20です。

XtNxrt3dTextFont XmFontList “fixed” CSG

テキスト・オブジェクトに含まれるテキストのフォントを指定します。このリソースは、XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_PIXELの時に使用されます。XtNxrt3dTextStrokeFontは、テキスト・オブジェクトがポイントまたはインデックスに貼り付けられている時にフォントを指定します。

XtNxrt3dTextForegroundColor char* NULL CSG

テキスト・オブジェクトの前景色を指定します。NULLであれば、グラフの前景色が使用されます。ストリングは、色名か、#で始まるRGB(赤、緑、青)構成の二桁の16進数です。色の指定については、2.7を参照してください。

モノクロのサーバでは、白以外の色を指定しますと黒に表示されます。

XtNxrt3dTextImage Xrt3dImage NULL CSG

テキスト・オブジェクトの背景に表示されるイメージを指定します。詳細はP38を参照してください。このリソースは、XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_PIXELの場合にのみ使用されます。

XtNxrt3dTextImageLayout Xrt3dImageLayout XRT3D_IMAGELAYOUT_CENTERED CSG

背景のイメージのレイアウトを指定します。有効な値は、XRT3D_IMAGELAYOUT_CENTEREDかXRT3D_IMAGELAYOUT_TILEDです。

XtNxrt3dTextImageMinimumSize Boolean False CSG

テキスト領域の最小サイズを背景のイメージによって決定させるかどうかを指定します。

XtNxrt3dTextLineShow Boolean True CSG

テキスト・オブジェクトと貼り付けポイントを結ぶ線を表示するかどうか指定します。この線は、1 pixel幅で、テキスト・オブジェクトの前景色を使用しています。

XtNxrt3dTextOffsetX Position 0 CSG
XtNxrt3dTextOffsetY Position 0 CSG

テキスト・オブジェクトの中央と貼り付けポイントの間のオフセット(pixel)を指定します。このリソースは、テキスト・オブジェクトがpixelに貼り付けられているか、またはXRT/3dが2D表示である場合に使用されます。

XtNxrt3dTextPlane int XRT3D_XZ_PLANE CSG

テキスト・オブジェクトの方向を指定します。テキスト・オブジェクトは、ユニット・キューブの一つの面と平行に描画され、常に左から右へテキストが表示されなければなりません。有効な値は、XRT3D_XZ_PLANEとXRT3D_YZ_PLANEです。このリソースは、2D表示では無視されます。

XtNxrt3dTextPrintFont char* NULL CSG

*Xrt3dDrawPS()*または*Xrt3dDrawCGM()*を使用してXRT/3dウィジェットをプリントする時、テキスト・オブジェクトに使用されるフォントを指定します。NULLの場合は、*Xrt3dDrawPS()*が自動的に一番近いPostScriptフォントを選択します。[1] *Xrt3dDrawCGM()*は、Roman Simplex ストローク・フォントを使用します。このリソースは、PostScriptまたはストローク・フォント[2] なら、どれでもセットすることができます。

XtNxrt3dTextShow Boolean True CSG

テキスト・オブジェクトを表示するかどうか指定します。

XtNxrt3dTextStrings char** NULL CSG

テキスト・オブジェクトに表示されるストリングのNULL終端リストを指定します。

XtNxrt3dTextStrokeFont Xrt3dStrokeFont XRT3D_SF_ROMAN_SIMPLEX CSG

テキスト・オブジェクトのテキストに使用されるストローク・フォントを指定します。このリソースは、XtNxrt3dTextAttachMethodがXRT3D_TEXT_ATTACH_POINTまたはXRT3D_TEXT_ATTACH_INDEXの時に使用されます。Appendix Eにある、ストローク・フォントのリストにあるフォントならどれでも使用できます。XtNxrt3dTextFontは、テキスト・オブジェクトがpixelに貼り付けられている時に使用されるXフォントを指定します。

XtNxrt3dTextStrokeSize Dimension 80 CSG

ストローク・フォントのサイズを指定します。サイズは、ユニット・キューブの大きさに対して千単位で測定され、範囲は0から1000です。ユニット・キューブの定義は、P15を参照してください。

B.3 継承されたリソース

XRT/3dテキスト・オブジェクトは、親のクラスからリソースを継承します。次の表は、リソースとそれを定義する親のクラスです。詳細は、MotifやXtの資料を参照してください。

Resource	Inherited From
XmNdestroyCallback	Object

[1] PostScriptフォントは、ほとんどのPSプリンタで使用可能な、35のPostScript Fontから選択されます。(P167の図51参照)

[2] 有効なストローク・フォントのリストは、Appendix Eにあります。

C

手続きと方法

XRT/3dの手続きと方法がアルファベット順に解説されています。“(source)”とある手続きには、ソースコードが提供されています。ソースは、*$XRTHOME/src/3d/util*にあります。ソース・ルーチンを変更する時は、他のディレクトリにコピーしてください。(名前を変更しない場合は、XRT/3dの前にリンクしてください。）

Xrt3dArgValToDouble()

`XtArgVal`をdoubleに変換します。*Xrt3dDoubleToArgVal()*と反対の手続きです。

```
double
Xrt3dArgValToDouble(
  XtArgVal    value
  )
```

Xrt3dComputeZValue()

グリッドまたはポイントにある値をインタラクティブに新しい値へドラッグする時に使用します。

```
double
Xrt3dComputeZValue(
  Widget  graph,
  int     index1, index2,
  int     pixel_x, pixel_y
  )
```

*graph*はXRT/3dウィジェットのハンドルです。グリッド・データでは、*index1*と*index2*は変更されるグリッド・セルを指します。ポイント・データでは、*index1*はシリーズ、*index2*は変更されるポイントの番号を指します。 *pixel_x*と*pixel_y*はドラッグされたポイントの新しい位置です。もし、(*index1*, *index2*)にあるグリッド値またはポイントが穴(hole)であれば、*Xrt3dComputeZValue()*は、そのグリッドのhole値を返します。*feedback3d*のデモ・プログラムはこの手続きを使用していますので参照してください。

Xrt3dContourStylesFromFile()

テキストファイルから読み込まれた情報を使用して、Xrt3dContourStyles構造体のポインタのNULL終端配列を作成します。その結果は、XtNxrt3dContourStylesの設定に使用できます。このルーチンで読み込まれるファイルのファイル名は、リソース・ファイルでXtNxrt3dContourStylesをセットする時に使用できます。

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

*filename*は、データ・ファイルのパス名です。(環境変数がサポートされています。)*Xrt3dContourStylesFromFile()*が、エラーに遭うと、NULLを返し、エラー・メッセージを*errbuf*に書き込みます。*errbuf*がNULLであれば、メッセージは、*stderr*に書かれます。*errbuf*は最低100 byte割り当てます。

"!"から始まる行はコメントとして扱われ、無視されます。他の行は、一つの等高線スタイルを定義し、次のようなフォーマットになります。

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

サンプルの等高線スタイルのファイルが、*$XRTHOME/src/3d/samples/100gray.col.*にあります。下記は、ファイル例です。

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

Xrt3dDataCopy() (source)

提供されたデータ・セットのコピーを作成します。

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

Xrt3dDataShaded() (source)

既存の構造体から新しいXrt3dData構造体を作成し、元のデータ・セットのshaded-relief map(影付の起伏図)でセットします。これは、4Dデータ・セットとして元のデータ上に重ね合わせる場合に適しています。

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

規則的、不規則的なグリッドのどちらもデータのソースにすることができます。

この手続きは、元のサーフェスの各面の反射率を計算し、その値を新しいグリッドに記憶します。このグリッドは、grey-scaleの等高線スタイルで表示させると、元のデータのshaded-relief mapと似ています。光源は、*sweep*(東西南北の方向)と*rise*(地平線からの角度)の二つの方角で定義される位置にあります。(地平線とみなされる)XY面の座標に対するZ値の割合は、*scale*パラメータで調節します。*scale*が0であれば、面の傾斜の平均からデフォルト値が計算されます。サーフェス全体の光の基本量は、*ambient*と呼ばれ、光の明るさは*intensity*で調節されます。*ambient*と*intensity*の値は、0から1の間です。

Xrt3dDataSmooth()(source)

各ポイント(グリッドの交差点)を中心に、その周囲のポイントとの平均を計算する関数をグリッドに渡します。その結果、データ・セットが滑らかに見えるようになります。

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

規則的、不規則的なグリッドのどちらもデータのソースにすることができます。*center_weight*は、0から1の間です。この手続きは、中心点とその周囲の全てのポイントを使用する9ポイントのフィルタです。新しい値は、元の値とその周囲のポイントとの平均です。平均化する時の、中心点に対する周囲のポイントの比重は、*center_weight*の値と中心点からの距離によって計算されます。

Xrt3dDataUpdateDataValue() (source)

Xrt3dData構造体のデータ値を新しい値に変更します。規則的、不規則的なグリッド及びポイント・データに使用できます。この手続きは、Z値のみを置き換えます。

```
int
Xrt3dDataUpdateDataValue(
  Xrt3dData   *data,
  double  oldValue,
  double  newValue
  )
```

*oldValue*はアップデートされる値、*newValue*は新しい値です。*Xrt3dDataUpdateDataValue()*は、成功時は1、失敗時は0を返します。

Xrt3dDataWindow() (source)

既存の構造体から新しいXrt3dData構造体を作成します。新しい構造体は、元のデータのサブセットを抽出し、指定に従って再計算されます。

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

規則的、不規則的なグリッドのどちらもデータのソースにすることができます。

新しいデータ・セットは、規則的なグリッドのデータセットで、起点が(*xstart, ystart*)、終点が(*xend, yend*)の領域に、*x*方向に*numx*本の線、*y*方向に*numy*本の線があります。ソースのグリッドで定義されている領域外の部分は、holeでセットされます。*interp*は、キュービック(XRT3D_INTERP_CUBIC_SPLINE)またはリニア(XRT3D_INTERP_LINEAR_SPLINE)のスプライン補間を指定します。

Xrt3dDestroyData()

Xrt3dData構造体を破棄します。

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

*data*は破棄される構造体です。*all*がTrueであると、データ値に割り当てられていたスペースは解放されます。

このデータ構造体は、その構造体を使用しているウィジェットから切り離されます。

Xrt3dDistnIndex()

分布表(disbrituion table)で指定された値に対するエントリーのインデックスを計算します。返された値は、*Xrt3dGetNthContourStyle()*を呼び出して、*value*に使用されている等高線スタイルのアクセスに使用することができます。

```
int
Xrt3dDistnIndex(
  Widget  graph,
  double  value
  )
```

*graph*は、グラフのハンドルです。

Xrt3dDistnTableFromFile()

テキスト・ファイルから読み込まれた情報を使用して、Xrt3dDistnTable構造体のポインタを作成します。その結果は、XtNxrt3dDistnTableリソースの設定に使用できます。このルーチンで読み込まれるファイルのファイル名は、リソース・ファイルでXtNxrt3dDistnTableをセットする時に使用できます。

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

*filename*は、データ・ファイルのパス名です。環境変数がサポートされています。*Xrt3DistnTableFromFile()*がエラーに遭うと、NULLを返し、エラー・メッセージを*errbuf*に書き込みます。*errbuf*がNULLであれば、メッセージは*stderr*に書かれます。*errbuf*には最低100 byte割り当てます。

"!"から始まる行はコメントとして扱われ、無視されます。他の行は、分布表の一つのエントリーを定義します。

下記は、ファイルの例です。

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

Xrt3dDoubleToArgVal()

double値を`XtArgVal`に変換し、XRT/3dの浮動小数点リソースの設定に使用します。

```
XtArgVal
Xrt3dDoubleToArgVal(
  double  value
  )
```

Xrt3dDrawCGM()

グラフのイメージをCGM(Computer Graphics Metafile)フォーマット(CGMバージョン1)に出力します。生成されたファイルはCGMバージョン2または3とも互換性があります。CGMファイルは、CGMプロッタやドキュメントにインポートすることができます。CGMファイルは、グラフが画面上で部分的に隠れていても、グラフの完全なイメージを持ちます。

```
int
Xrt3dDrawCGM(
  XtXrt3dWidget    graph,
  FILE         *fp,
  char         *msg,
  double       metric,
  Xrt3dStrokeFont  header_font,
  Xrt3dStrokeFont  footer_font,
  Xrt3dStrokeFont  legend_font
  )
```

*graph*は出力されるグラフのハンドルです。*fp*は既に開かれ書き込みの準備ができているファイルを指しています。*Xrt3dDrawCGM()*が問題に遭うと、メッセージを*msg*に返します。(*msg*は最低100 byte割り当てます。) *msg*が`NULL`であると、エラー・メッセージは*stderr*に書かれます。*Xrt3dDrawCGM()*は、失敗時には0、成功時には1を返します。

*metric*は出力のサイズを調節します。0.0であれば、出力デバイスがイメージのサイズをコントロールします。0以外の場合は、*metric*がミリメートルに対するデバイスのユニット数を指定します。例えば、300x600 pixelのグラフで*metric*を0.5に設定すると、150x300mmのCGMファイルが生成されます。

XフォントはCGMの文字セットに直接にマップできないので、*header_font*、*footer_font*、*legend_font*は、それぞれヘッダー、フッター、凡例に使用される代替のストローク・フォントを指定します。*Xrt3dDrawCGM()*は、Xフォントの単位を使用してフォントのサイズを決定します。(`Xrt3dStrokeFonts`の定義は、Appendix Eにあります。)

次のコードは、*Xrt3dDrawCGM()*の使用例です。

```
char msg[100];
FILE *fp;
Xrt3dDrawCGM(mygraph, fp, msg,0.0,     /* Don't set scale */
  XRT3D_SF_ITALIC_COMPLEX,          /* Header font */
  XRT3D_SF_ROMAN_DUPLEX,       /* Footer font */
  XRT3D_SF_ROMAN_SIMPLEX);          /* Legend font */
```

この機能は、XRT/3d 3.0.0ではサポートされておらず、前のバージョンとの互換性を保つために残っていますので、XRT/3d 2.1.1より後のバージョンでは動作が保証されていません。

Xrt3dDrawPS()

グラフのEPS(encapsulated PostScript、EPSF-2.0)をファイルに書きます。なるべく、*Xrt3dOutputPS()*より*Xrt3dDrawPS()*の方を使用してください。*Xrt3dOutputPS()*は、PostScriptの*image*オペレータを使って画面のpixelを出力しますが、*Xrt3dDrawPS()*はデバイスに依存しないPostScriptオペレータを使用してネイティブのPostScriptファイルを出力します。*Xrt3dDrawPS()*によって生成されたファイルは、PSプリンタの解像度を完全に利用して印刷できます。

ウィジェットは、この手続きを呼び出す前に、realizeされなければなりません。(但し、ディスプレイにマップされている必要はありません。)

```
int
Xrt3dDrawPS(
  Widget  graph,
  FILE    *fp,
  char    *msg,
  int     inches,
  double  paper_width, paper_height, margin,
  int     landscape,
  double  x_offset, y_offset, width, height,
  char    *header_font, *footer_font,
  char    *legend_font,
  int     fill_background,
  int     smart_mono,
  int     showpage
  )
```

*Xrt3dDraw()*は失敗時には0、成功時には1を返します。*graph*は出力されるグラフのハンドルです。*fp*は既にオープンされ書き込みができるファイルを指していなければなりません。*Xrt3dDrawPS()*が問題に遭遇した場合は、*msg*(最低100 byte割り当てます)にメッセージを返します。*msg*がNULLであれば、メッセージは*stderr*に書かれます。

*inches*がTrueであれば、用紙の幅、高さ、マージンに与えられた値と、イメージ・オフセットの幅と高さに与えられた値はインチで解釈されます。それ以外の場合は、センチ・メートルが使用されます。

paper_width、*paper_height*と*margin*は必ず指定してください。*margin*はページの周りの余白の幅を定義します。これはページの端まで印刷できないプリンタには必ず使用します。

*landscape*がTrueであれば、ページが横向きに印刷されます。

*x_offset*と*y_offset*は、用紙の左下角に対応するイメージの左下角のオフセットを指定します。例えば、用紙の下から2インチ、左から3インチの角へイメージの左下角を合わせる時は、*inches*を`True`にし、*x_offset*を3.0、*y_offset*を2.0に設定します。

*width*と*height*はイメージのサイズを指定します。両方とも0.0に設定するとイメージは、元の比率を保ちながら、最大限に印刷されます。もし二つのパラメータの内、一つだけが0以外であれば、その値が使用されますが、元の比率は保たれます。二つとも0以外の場合は、その二つの値が適用され比率が変更されます。(イメージは指定されたサイズに合うよう幅と高さが調整されます。)

ヘッダー、フッター、凡例に使用されるPostScriptフォントはフォント名で指定します。`NULL`が指定されている場合は、*Xrt3dDrawPS()*は、画面で使用されているXフォントについてXサーバに問い合わせ、35の標準のLaserWriter IIフォントのリストの中から一番近いフォントを選択します。

全てのXサーバがフォントについて同じ情報を持っているとは限らないので、異なったXサーバでアプリケーションが実行された場合に、異なったPostScriptフォントが得られることがあります。*Xrt3dDrawPS()*を使用する時は、フォントをハードコーディングすることができます。(例: *header_font*を`"Times-Roman,Ryumin-Light"`に指定)

```
Times-Roman             Helvetica               NewCenturySchlbk-Roman
Times-Bold              Helvetica-Bold          NewCenturySchlbk-Bold
Times-Italic            Helvetica-Oblique       NewCenturySchlbk-Italic
Times-BoldItalic        Helvetica-BoldOblique   NewCenturySchlbk-BoldItalic

Palatino-Roman          Bookman-Light           Helvetica-Narrow
Palatino-Bold           Bookman-Demi            Helvetica-Narrow-Bold
Palatino-Italic         Bookman-LightItalic     Helvetica-Narrow-Oblique
Palatino-BoldItalic     Bookman-DemiItalic      Helvetica-Narrow-BoldOblique

Courier                 AvantGarde-Book         Symbol
Courier-Bold            AvantGarde-Demi         ZapfChancery-MediumItalic
Courier-Oblique         AvantGarde-BookOblique  ZapfDingbats
Courier-BoldOblique     AvantGarde-DemiOblique
```

図51　35の標準LaserWriter II フォント名

*fill_background*が`False`の場合、*Xrt3dDrawPS()*は背景を色付けしません。

*smart_mono*が`XRT3D_PS_COLOR_AS_IS`の場合、画面と同じ色がプリントされます。`XRT3D_PS_COLOR_MONO`の場合は色はgray shadeにマップされ、`XRT3D_PS_COLOR_AUTO`の場合は、出力デバイスがカラーかモノクロかを判断し[1]、その結果に従ってプリントされます。この引数は、モノクロのXサーバでは無視されます。

*showpage*が`True`の場合、*Xrt3dDrawPS()*は、EPSFファイルの終わりにPostScript *showpage*オペレータを付けます。同じファイル・ポインタを使用して*Xrt3dDrawPS()*へ幾つかの呼び出しを行い、最後の呼び出し以外全て*showpage*を`False`に設定することにより、1ページに幾つかのグラフを印刷することも可能です。

[1] SPARCprinter等のモノクロのプリンタは、間違ってカラー・プリンタと識別してしまいます。これらのプリンタでsmart_monoを使用するには、`XRT3D_PS_COLOR_MONO`にセットしてください。

下の例は、グラフを2.0cmのマージンで標準のA4用紙に、元の比率を保ちながら最大限に印刷します。全てのフォントはデフォルトです。

```
char msg[100];
FILE *fp;
Xrt3dDrawPS(mygraph, fp, msg,
 False,           /* Use cm */
 21.0, 29.7, 2.0,     /* 21.0x29.7cm, 2.0cm margin */
 False,           /* Portrait */
 0.0, 0.0, 0.0, 0.0,  /* Full Size */
 NULL, NULL, NULL,    /* Default Fonts */
 True,            /* Keep background */
 False,           /* Don't adjust */
 True);           /* Show the page */
```

Xrt3dDupContourStyles()

Xrt3dContourStyle構造体の配列を複写します。複写された配列のポインタが返されます。

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

Xrt3dDupDataStyles()

Xrt3dDataStyle構造体の配列を複写します。

```
Xrt3dDataStyle **
Xrt3dDupDataStyles(
      Xrt3dDataStyle **ds
      )
```

Xrt3dDupDistnTable()

Xrt3dDistnTable構造体を複写します。複写された表(table)のポインタが返されます。割り当てが失敗するとNULLが返されます。

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

Xrt3dDupStrings()

ストリングの配列を複写します。複写された配列のポインタが返されます。割り当てが失敗するとNULLが返されます。

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

Xrt3dDupValueLabels()

値のラベルのリストを複写します。

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

*vlabels*は、Xrt3dValueLabel型の構造体のポインタのNULL終端配列です。このリストのコピーが呼出し側に返されます。割り当てに失敗するとNULLが返されます。

Xrt3dDupXYColors()

XYColorsのリストを複写します。

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

*xycols*は、Xrt3dXYColor型の構造体のポインタのNULL終端配列です。このリストのコピーが呼出し側に返されます。割り当てに失敗するとNULLが返されます。

Xrt3dEnableImageType()

XRT/3dに新しいイメージの型を登録します。(PNGやJPEG等の)定義済みの型や、ユーザが定義した型を使用できます。

```
Boolean
XrtEnableImageType(
      XtPointer    class,
      XtPointer    functions
      )
```

*class*はウィジェット・クラス(xtXrt3dWidgetClass)。*funciton*はXrtPDSImageFuncs構造体のポインタです。詳細は、P106を参照してください。

Xrt3dFreeContourStyles()

Xrt3dContourStyle構造体の配列に使用されていたメモリを解放します。

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

Xrt3dFreeDataStyles()

Xrt3dDataStyle構造体の配列に使用されていたメモリを解放します。

```
void
Xrt3dFreeDataStyles(
      Xrt3dDataStyle   **ds
      )
```

Xrt3dFreeDistnTable()

Xrt3dDistnTable構造体に使用されていたメモリを解放します。

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

Xrt3dFreePropString()

*Xrt3dGetPropString()*で割り当てられたストリングに使用されていたメモリを解放します。

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

Xrt3dFreeStrings()

ストリングの配列に使用されていたメモリを解放します。

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

Xrt3dFreeValueLabels()

値ラベルのリストを解放します。

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

*vlabels*は、Xrt3dValueLabel型の構造体のポインタのNULL終端配列です。これらの構造体とポインタ配列に使用されていたメモリは解放されます。ウィジェットで現在使用されているXrt3dValueLabelのリストは解放させないでください。

Xrt3dFreeXYColors()

XYColorsのリストを解放します。

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

*xycols*は、Xrt3dXYColor型の構造体のポインタのNULL終端配列です。これらの構造体とポインタ配列に使用されていたメモリは解放されます。ウィジェットで現在使用されているXrt3dXYColorのリストは解放させないでください。

Xrt3dGetDistnTable()

ウィジェットで現在使用されている分布表(distribution table)のポインタを返します。

```
Xrt3dDistnTable *
Xrt3dGetDistnTable(
  Widget   graph
  )
```

この手続きは、返された表のスペースを割り当てます。これは、*Xrt3dFreeDistnTable()*を呼び出すことにより解放されます。失敗すると、NULLを返します。

Xrt3dGetImage()

グラフのXImageのポインタを返します。これは、ウィジェットのXImageにアクセスする必要のあるXlibレベルのプログラミングに便利です。

```
XImage *
Xrt3dGetImage(
 Widget  graph
 )
```

*graph*は出力されるグラフのハンドルです。*Xrt3dGetImage()*は失敗時にNULLを返します。グラフが画面上で部分的にしか表示されていない場合でもXImageは完成します。

Xrt3dGetNthContourStyle()

等高線スタイルのポインタを返します。

```
Xrt3dContourStyle *
Xrt3dGetNthContourStyle(
 Widget  graph,
 int index,
 Boolean used
 )
```

*graph*は、グラフのハンドルです。*index*(0-based)は、どの等高線スタイルを返すか指定します。実際に使用される等高線スタイルのサブセットは、分布(distribution)のレベルの数(*nlevels*)に依存します。*used*をTrueにセットすると、実際に使用されているスタイルのセットから、等高線スタイルを検索します。Falseにすると、現在定義されている全てのスタイルのセットから、等高線スタイルを検索します。この方法で返されたポインタはread-onlyです。

Xrt3dGetNthDataLabel()

データ・ラベルのポインタを返します。

```
char *
Xrt3dGetNthDataLabel(
 Widget      graph,
 Xrt3dAxis   axis,
 int     index
 )
```

*graph*は、グラフのハンドルです。*axis*は、ラベルを取得する軸を指定します。*index*は、要求されたラベルの(0-indexed)数です。軸にデータ・ラベルがない時、または、*index*が範囲外の時は、*Xrt3dGetNthDataLabel()*は、NULLを返します。

Xrt3dGetNthDataStyle()

グラフによって現在使用されている*n*番めのデータ・スタイルのポインタを返します。

```
Xrt3dGetNthDataStyle(
     Widget  graph,
     Int index
     )
```

*graph*はグラフのハンドルです。*index*(0から開始)は、どのデータ・スタイルを返すか指定します。*index*が現在定義されているデータ・スタイルの数より大きい場合は、*index*のデフォルトのデータ・スタイルが返されます。この方法で返されたポインタは、read-onlyです。

Xrt3dGetNthFooterString()

現在グラフで使用されている*n*番めのフッター・ストリングのポインタを返します。

```
char *
Xrt3dGetNthFooterString(
  Widget  graph,
  int     index
  )
```

*graph*はグラフのハンドルです。*index*(0から開始)は、どのフッター・ストリングを返すか指定します。*index*が現在定義されているフッター・ストリングの数より大きい場合は、*index*のデフォルトのフッター・ストリングが返されます。この方法で返されたポインタは、read-onlyです。

Xrt3dGetNthHeaderString()

現在グラフで使用されている*n*番めのヘッダー・ストリングのポインタを返します。

```
char *
Xrt3dGetNthHeaderString(
  Widget  graph,
  int     index
  )
```

*graph*はグラフのハンドルです。*index*(0から開始)は、どのヘッダー・ストリングを返すか指定します。*index*が現在定義されているヘッダー・ストリングの数より大きい場合は、*index*のデフォルトのヘッダー・ストリングが返されます。この方法で返されたポインタは、read-onlyです。

Xrt3dGetNthLegendString()

現在グラフで使用されている*n*番めの凡例のストリングのポインタを返します。

```
char *
Xrt3dGetNthLegendString(
  Widget  graph,
  int     index
  )
```

*graph*はグラフのハンドルです。*index*(0から開始)は、どの凡例のストリングを返すか指定します。*index*が現在定義されている凡例のストリングの数より大きい場合は、*index*のデフォルトの凡例ストリングが返されます。この方法で返されたポインタは、read-onlyです。

Xrt3dGetPropString()

グラフ・リソースの現在の値をストリングとして取り出します。

```
Boolean
Xrt3dGetPropString(
  Widget  graph,
  int     resource,
  char    **str    /* Returned */
  )
```

*graph*は、グラフのハンドルです。*resource*は、(P205にあるクロス・プラットフォーム名を使用した)リソースを指定します。*str*は、リソース値を表す、返されたストリングのポインタです。*str*は、使用後、*Xrt3dFreePropString()*を使用して解放してください。

Xrt3dGetValueLabel()

提供されたラベルに一番近い値を持つ、値ラベルのポインタを返します。

```
Xrt3dValueLabel *
Xrt3dGetValueLabel(
  Widget       graph,
  Xrt3dAxis    axis,
  Xrt3dValueLabel  *vlabel
  )
```

*graph*は、グラフのハンドルです。*axis*は、ラベルを取得する軸を指定します。*vlabel*は、セットされた*value*を持つ`Xrt3dValueLabel`構造体を指します。要求された値に一番近く位置する値ラベルのポインタが返されます。この*axis*に値ラベルがない場合は、*Xrt3dGetValueLabel()*は、`NULL`を返します。

Xrt3dGetValues()

リソースの現在の値を取り出します。

```
void
Xrt3dGetValues(
  Widget  graph,
  ...,
  NULL
  )
```

*graph*は、グラフのハンドルです。...とある場所には、*resource-value*のペアが来ます。*resource*は、(P205のリストにある、`Xrt3dProperty`クロス・プラットフォーム名として指定された)リソースを指します。*value*は、返されたリソース値を持つ変数のポインタです。*value*のデータ型は、リソースのデータ型と同じでなければなりません。`double`型のリソースにアクセスする時は、*Xrt3dArgValToDouble()*は使用しないでください。このリストは、`NULL`で終了します。

Xrt3dGetXYColor()

棒グラフの色のポインタを返します。

```
char *
Xrt3dGetXYColor(
  Widget  graph,
  int xindex, yindex
  )
```

*graph*は、グラフのハンドルです。*xindex*と*yindex*は、色を取り出す棒または、棒のセットを指定します。

Xrt3dInsertNthDataStyle()

*n*番めに来る新しいデータ・スタイルを挿入します。

```
int
Xrt3dInsertNthDataStyle(
      Widget       graph,
      int      index
      Xrt3dDataStyle   *ds
      )
```

*graph*はグラフのハンドル、*index*(0から開始)は、新しいデータ・スタイルのインデックスです。*index*と同じかそれより大きいインデックスのデータ・スタイルは全て繰り上げられます。*ds*は新しいデータ・スタイルを持つXrt3dDataStyle構造体を指し、XRT/3dは、*ds*で指されているデータ・スタイルのコピーを作成します。

この方法を使用すると、副作用でXtNxrt3dDataStyleUseDefaultがFalseにセットされます。

Xrt3dMakeDataFromFile()

Xrt3dData構造体を割り当て、テキスト・ファイルから読み込まれたデータをロードします。*Xrt3dSaveDataToFile()*も参照してください。

```
Xrt3dData *
Xrt3dMakeDataFromFile(
  char*filename,
  char*errbuf
  )
```

*filename*は、データ・ファイルのパス名です。環境変数を使用することもできます。*Xrt3dMakeDataFromFile()*がエラーに遭遇すると、NULLを返し、*errbuf*にエラー・メッセージを書き込みます。*errbuf*がNULLであれば、メッセージは*stderr*に書かれます。*errbuf*には最低100 byte割り当てます。

"!"で始まる行は、コメントとして扱われ、無視されます。最初のコメントでない行は、データ型で始まり、GRID、IRGRIDまたはPOINTのどれかです。GRIDまたはIRGRIDの場合、次には、XとYの方向のポイントの数を指定する二つの整数(≥1)が来ます。POINTの場合は、シリーズの数を指定する一つの整数(>1)が来ます。その次は、データの穴(hole)の値を指定する浮動小数点数です。

GRIDが指定されている場合、次の二つの浮動小数点数は、XとYの方向のグリッドの間隔(grid step)を指定します。その次は、XとYの原点の値を示す浮動小数点数が来ます。最後は、Yの値が増加する順に並んだサーフェス・データです。(最初にX=originの時のY値、そしてX=(x origin + x grid step)の時のY値、etc)

GRIDデータ・ファイルの例:

```
! 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
. . .
```

IRGRIDが指定された場合、*xgrid*と*ygrid*の値のリストが、原点、*xstep*、*ystep*の値の代わりに提供されます。IRGRIDデータ・ファイルの例:

```
! 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
. . .
```

POINTが指定された場合、シリーズ・データのリストが提供されます。各シリーズは、シリーズ内のポイント数を表す整数と(x, y, z)値のリストで指定されます。(下記例参照)

```
!Point data has 3 series.
!Holes have value -1000
POINT 3
-1000
!Series 1 has 5 points. 15 (x, y, z) values follow.
5
0.75 0.50 0.30
0.45 0.50 0.30
0.75 0.40 0.30
0.75 0.50 0.40
0.45 0.40 0.40
!Series 2 has 3 points. 9 (x, y, z) values follow.
3
1.2 1.3 1.4
1.5 1.6 1.7
1.8 1.9 2.0
!Series 3 has 4 points. 12 (x, y, z) values follow.
4
-3.0 -2.0 -1.0
0.0 1.0 2.0
2.0 1.0 0.0
-1.0 -2.0 -3.0
```

Xrt3dMakeGridData()

空のXrt3dData構造体を規則的なグリッド・データに割り当てる手続きです。スペースを不規則的なグリッド・データに割り当てる場合は、*Xrt3dMakeIrGridData()*を使用します。

```
Xrt3dData *
Xrt3dMakeGridData(
 int numx, numy,
 double  noval,
 double  xstep, ystep,
 double  xorig, yorig,
 Boolean all
 )
```

*numx*と*numy*は、グリッドのXとY方向のエレメントの数を指定し、常に≥1です。*noval*は、XRT/3dがデータの穴(hole)として扱う値を定義します。*xstep*と*ystep*は、XとY方向のグリッドの間隔を指定します。*xorig*と*yorig*は、原点を定義します。*all*がTrueであれば、データ値のスペースも割り当てられます。それ以外の場合は、構造体だけが割り当てられ、渡された値がその構造体にセットされます。

Xrt3dMakeIrGridData()

空のXrt3dData構造体を不規則的なグリッド・データに割り当てる手続きです。スペースを規則的なグリッド・データに割り当てる場合は、*Xrt3dMakeGridData()*を使用します。

```
Xrt3dData *
Xrt3dMakeIrGridData(
 int numx, numy,
 double  noval,
 Boolean all
 )
```

*numx*と*numy*は、グリッドのXとY方向のエレメントの数を指定し、常に≥1です。*noval*は、XRT/3dがデータの穴(hole)として扱う値を定義します。*all*がTrueであれば、データ値、*xgrid*と*ygrid*の配列のスペースが割り当てられます。それ以外の場合は、構造体だけが割り当てられ、渡された値がその構造体にセットされます。

Xrt3dMakePointData()

空のXrt3dData構造体をポイント・データに割り当てる手続きです。Xrt3dData構造体のメモリが、*nsereis* Xrt3dPointSeriesの配列と一緒に割り当てられます。

```
Xrt3dData*
Xrt3dMakePointData(
      int nseries,
      int npoints,
      double  noval,
      Boolean all
      )
```

*nseries*は割り当てるXrt3dData構造体の数を指定し、常に≥1です。*all*がTrueであれば、各シリーズに対して*npoints* Xrt3dPoint3Dが割り当てられます。Xrt3dPoint3Dは次のように定義されます。

```
typedef struct {
      double  x, y, z;
} Xrt3dPoint3D;
```

Xrt3dMakePointDataSeries()

渡されたXrt3dPointSeriesに記憶される*npoints* Xrt3dPoint3D構造体の配列を割り当て、その前に使用されていたポイントを解放します。Xrt3dPoint3D構造体は次のように定義されます。

```
typedef struct {
     double x, y, z;
} Xrt3dPoint3D;

Boolean
Xrt3dMakePointDataSeries(
     Xrt3dPointSeries *pseries,
     int     npoints,
     )
```

*pseries*はXrt3dPointSeries構造体のポインタです。*npoints*は割り当てるポイントの数です。

Xrt3dMap()

pixel座標をグラフ座標にマップします。

```
Xrt3dRegion
Xrt3dMap(
 Widget      graph,
 Position    pix_x, pix_y,
 Xrt3dMapResult   *map
 )
```

アプリケーションに使用され、*graph*をハンドルとするグラフで、*pix_x*と*pix_y*に対応するグラフ座標を決定し、結果を*map*に返します。これは、イベント処理手続きに使用されます。

Xrt3dMrmInitialize()

MrmにXRT/3dウィジェット・クラスに関する情報を伝えます。UILアプリケーションに使用されます。このルーチンは、*MrmInitialize()*の後に呼び出してください。

```
void
Xrt3dMrmInitialize(void)
```

Xrt3dOutputPS()

グラフのEPSF(encapsulated PostScript)をファイルに出力します。PostScriptファイルは、PostScriptプリンタに送ったり、他のPostScriptファイルへのインポートに適しています。ファイルは、EPSF-2.0フォーマットに準拠しています。この手続きは、"bitmap PostScript" ファイルを作成しますが、このファイルはプリント・デバイスの解像度を有効に活用して印刷しないので、通常は、*Xrt3dDrawPS()*を使用してください。

```
int
Xrt3dOutputPS(
 Widget  graph,
 FILE*fp,
 Boolean color,
 char*msg
 )
```

*graph*は出力されるグラフのハンドルです。*fp*は、既にオープンされ書き込みできるファイルを指していなければなりません。*color*が`True`なら、PostScriptの*colorimage*が作成されますが、それ以外の場合は、モノクロのPostScript *image*が作成されます。

*Xrt3dOutputPS()*が問題に遭うと、*msg*(最低100 byte割り当てます)にメッセージを返します。*msg*が`NULL`であれば、エラー・メッセージは*stderr*に書かれます。*Xrt3dOutputPS()*は失敗時に0を返し、成功時には1を返します。グラフが画面上で部分的に表示されていない場合でもPostScriptファイルは完成します。カラーのグラフのPostScriptファイルはサイズが大きくなりますので、このルーチンを呼び出す前に、十分なスペースがあるかを確認してください。

Xrt3dOutputXwd()

グラフ・イメージのX Window Dumpをファイルに出力します。

```
int
Xrt3dOutputXwd(
  Widget  graph,
  FILE*fp,
  char*msg
  )
```

*graph*は出力されるグラフのハンドルです。*fp*は、既にオープンされ書き込みできるファイルを指していなければなりません。*Xrt3dOutputXwd()*が問題に遭うと、*msg*(最低100 byte割り当てます)にメッセージを返します。*msg*が`NULL`であれば、エラー・メッセージは、*stderr*に書かれます。*Xrt3dOutputXwd()*は失敗時に0を返し、成功時には1を返します。グラフが画面上で部分的に表示されていない場合でもxwdファイルは完成します。

Xrt3dPick()

与えられたpixel座標に一番近いデータをピックします。

```
Xrt3dRegion
Xrt3dPick(
  Widget       graph,
  Position     pix_x, pix_y,
  Xrt3dPickResult  *pick
  )
```

アプリケーションに使用され、*graph*をハンドルとするグラフで、どのデータ(グリッド・ポイント・インデックス)がpixel(*pix_x*、*pix_y*)に一番近く表示されているかを決定し、その結果を*pick*に返します。イベント処理手続きに使用されます。

Xrt3dRemoveNthDataStyle()

*n*番めのデータ・スタイルを返します。

```
int
Xrt3dRemoveNthDataStyle(
      Widget  graph,
      int index,
      )
```

*graph*はグラフのハンドル、*index*(0から開始)は削除されるデータ・スタイルのインデックスです。*index*より大きいインデックスのデータ・スタイルは全て繰り下げられます。この手続きは、成功時には1、失敗時には0を返します。

Xrt3dResetContourStyles()

XRT/3dにデフォルトの等高線スタイルを使用するように指定します。

```
void
Xrt3dResetContourStyles(
  Widget  graph,
  )
```

*graph*は、グラフのハンドルです。

Xrt3dSampleColorFunc()

XtNxrt3dColorFuncに使用できるファンクションの例を提供します。XPMが一番近い色と適応させる時に使用するアルゴリズムの応用です。これのソースコードは、*$XRTHOME/src/3d/util/x_color_func.c*にあります。

```
unsigned long
Xrt3dSampleColorFunction(
      Display     *display,
      int     screen,
      Colormap    colormap,
      Xcolor      *color,
      Boolean     do_free
      )
```

Xrt3dSaveDataToFile()

Xrt3dData構造体にあるデータを、*Xrt3dMakeDataFromFile()*に適するフォーマットでテキスト・ファイルに書きます。

```
int
Xrt3dSaveDataToFile(
  Xrt3dData   *data,
  char    *filename,
  char    *errbuf
  )
```

*data*は有効なXrt3dData構造体を指していなければなりません。*filename*はデータ・ファイルのパス名です。環境変数を使用することもできます。エラーが起きるとNULLを返し、メッセージは*errbuf*に書かれます。*errbuf*がNULLであれば、メッセージは*stderr*に書かれます。(*errbuf*には最低100 byte割り当てます。)この手続きは、成功時には1を失敗時には0を返します。

Xrt3dSaveImageAsJpeg()

グラフをJPEGに出力します。これを使用する前に、JPEGを*Xrt3dEnableImageType()*を使用してXRT/3dに登録してください。

```
Boolean
Xrt3dSaveImageAsJpeg(
      Widget  graph,
      String  filename,
      String  msg
      )
```

*graph*は出力されるグラフのハンドル、*filename*はイメージが書かれるファイルの名前です。*Xrt3dSaveImageAsJpeg()*が問題に遭遇すると*msg*(最低100byte)にメッセージを返します。*msg*がNULLの場合は、エラー・メッセージは*stderr*に書かれます。*Xrt3dSaveImageAsJpeg()*は成功時にはTrueを返します。

Xrt3dSaveImageAsPng()

グラフをPNGに出力します。これを使用する前に、PNGを*Xrt3dEnableImageType()*を使用してXRT/3dに登録してください。

```
Boolean
Xrt3dSaveImageAsPng(
     Widget  graph,
     String  filename,
     String  msg
     )
```

*graph*は出力されるグラフのハンドル、*filename*はイメージが書かれるファイルの名前です。*Xrt3dSaveImageAsPng()*が問題に遭遇すると*msg*(最低100byte)にメッセージを返します。*msg*がNULLの場合は、エラー・メッセージは*stderr*に書かれます。*Xrt3dSaveImageAsPng()*は成功時にはTrueを返します。

Xrt3dSaveImageGeneric()

グラフをそのイメージ型のまま出力します。既に登録されているXPMやXWD以外の型は、この手続きを使用する前に、*Xrt3dEnableImageType()*を使用してXRT/3dに登録してください。

```
Boolean
Xrt3dSaveImageGeneric(
     Widget  graph,
     int image_type
     String  filename,
     String  msg
     )
```

*graph*は出力されるグラフのハンドル、*image_type*はイメージ型、*filename*はイメージが書かれるファイルの名前です。*Xrt3dSaveImageGeneric()*が問題に遭遇すると*msg*(最低100byte)にメッセージを返します。*msg*がNULLの場合は、エラー・メッセージは*stderr*に書かれます。*Xrt3dSaveImageGeneric()*は成功時にはTrueを返します。

Xrt3dSetNthContourStyle()

*n*番めの等高線スタイルをセットします。

```
void
Xrt3dSetNthContourStyle(
 Widget          graph,
 int         index,
 Xrt3dContourStyle    *cs,
 Boolean         used
 )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どの等高線スタイルが変更されるかを指定します。*cs*は、新しい等高線スタイルを含むXrt3dContourStyle構造体を指します。実際に使用される等高線スタイルのサブセットは、分布(distribution)のレベル数(*nlevels*)に依存します。

*used*が`True`であれば、*idnex*はその時に使用されているスタイルだけを参照します。*used*が`False`であれば、*index*は、定義されている等高線スタイルの配列全体を参照します。XRT/3dは、*cs*で指されている等高線スタイルのコピーを作成します。

Xrt3dSetNthDataLabel()

*n*番めのデータ・ラベルをセットします。

```
void
Xrt3dSetNthDataLabel(
 Widget      graph,
 Xrt3dAxis   axis,
 int     index,
 char    *label
 )
```

*graph*は、グラフのハンドルです。*axis*は、ラベルが付けられる軸を指定します。*index*は、新しいラベルの(0-indexed)数です。*label*は新しいラベルの値です。*index*が使用されているラベルの数より大きい場合、その間にあるラベルには、空のストリングが割り当てられます。

Xrt3dSetNthDataStyle()

*n*番めのデータ・スタイルをセットします。

```
void
Xrt3dSetNthDataStyle(
     Widget      graph,
     int     index,
     Xrt3dDataStyle   *ds
     )
```

*graph*はグラフのハンドル、*index*(0から開始)はどのデータ・スタイルを変更するか指定します。*ds*は新しいデータ・スタイルを持つ`Xrt3dDataStyle`構造体を指します。XRT/3dは、*ds*によって指されているデータ・スタイルのコピーを作成します。

この方法を使用すると、副作用で`XtNxrt3dDataStylesUseDefault`が`False`にセットされます。

Xrt3dSetNthFooterString()

*n*番めのフッター・ストリングをセットします。

```
void
Xrt3dSetNthFooterString(
  Widget  graph,
  int     index,
  char    *s
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのフッター・ストリングを変更するか指定します。*s*は、使用される新しいストリングです。XRT/3dは、*s*で指されている文字のコピーを作成します。

Xrt3dSetNthHeaderString()

*n*番めのヘッダー・ストリングをセットします。

```
void
Xrt3dSetNthHeaderString(
  Widget  graph,
  int     index,
  char    *s
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どのヘッダー・ストリングを変更するか指定します。*s*は、使用される新しいストリングです。XRT/3dは、*s*で指されている文字のコピーを作成します。

Xrt3dSetNthLegendString()

*n*番めの凡例のストリングをセットします。

```
void
Xrt3dSetNthLegendString(
  Widget  graph,
  int     index,
  char    *s
  )
```

*graph*は、グラフのハンドルです。*index*(0から開始)は、どの凡例のストリングを変更するか指定します。*s*は、使用される新しいストリングです。XRT/3dは、*s*で指されている文字のコピーを作成します。

Xrt3dSetPropString()

グラフのリソースをストリングで表される値にセットします。

```
Boolean
Xrt3dSetPropString(
  Widget      graph,
  Xrt3dProperty   resource,
  char        *str
  )
```

*graph*は、グラフのハンドルです。*resource*は、(P205のリストにある、クロス・プラットフォーム名を使用した)XRT/3dのリソースを指定します。*str*は、セットされるリソースの値を持つストリングです。(構文はAppendix Gにあるリソース・ファイルと同じです。)

Xrt3dSetReverseConverters()

*XtConvertAndStore()*と使用される逆変換プログラムを登録します。逆変換プログラムは、アプリケーションに、人間が読める形のリソース値を提供します。

```
void
Xrt3dSetReverseConverters(void)
```

Xrt3dSetValueLabel()

値ラベルを定義、置き換えまたは、削除します。

```
void
Xrt3dSetValueLabel(
 Widget       graph,
 Xrt3dAxis    axis,
 Xrt3dValueLabel  *vlabel
 )
```

*graph*は、グラフのハンドルです。*axis*は、ラベルが付けられる軸を指定します。*vlabel*は、セットされた*value*と*label*を持つ`Xrt3dValueLabel`構造体を指します。*value*が、既に存在している値ラベルと同じであれば、新しいラベルが既存のラベルを置き換えます。新しいラベルが`NULL`であれば、既存のラベルは、削除されます。

(四捨五入の誤差を避けるため)変更または削除される値ラベルは、*Xrt3dGetValueLabel()*を使用して探してください。

Xrt3dSetValues()

グラフのリソースをセットします。

```
void
Xrt3dSetValues(
 Widget  graph,
 ...,
 NULL
 )
```

*graph*は、グラフのハンドルです。...とある場所には、*resource-value*のペアが来ます。*resource*は、(P205のリストにある、`Xrt3dProperty`クロス・プラットフォーム名として指定された)リソースを指します。*value*は、新しい値を指定します。*value*のデータ型は、リソースのデータ型と同じでなければなりません。`double`型のリソースにアクセスする時は、*Xrt3dDoubleToArgVal()*は使用しないでください。このリストは、`NULL`で終了します。

Xrt3dSetXYColor()

棒グラフの色を定義、置き換えまたは削除します。

```
void
Xrt3dSetXYColor(
 Widget  graph,
 int xindex, yindex,
 char*color
 )
```

*graph*は、グラフのハンドルです。*xindex*と*yindex*は、*color*が適用される棒または棒のセットを指定します。*xindex*が-1であれば、*color*は、*yindex*で定義されている列の棒全てに適用されます。*yindex*が-1であれば、*color*は、*xindex*で定義されている行の棒全てに適用されます。*xindex*と*yindex*の両方が-1であれば、*color*は、グラフの全ての棒に適用されます。色のリクエストは、指定された順に適用されます。例えば、青を(-1, 3)に、次に赤を(2, 3)に、と指定されている場合、3列目の棒は全て青になり、*xindex2*の棒だけ赤になります。

XRT/3dは、XYColorsの内部リストに、(*xindex, yindex*)につき一つのエントリーしか持ちません。*Xrt3dSetXYColor()*が、*color*がNULLで呼ばれると、マッチしたエントリーがリストから削除されます。それ以外の場合、*Xrt3dSetXYColor()*のコールは、新しいエントリーをリストの最後に置き、二重になったものを削除します。

この手続きは、XtNxrt3dTypeがXRT3D_TYPE_BAR、XtNxrt3dDrawShadedがTrue、XtNxrt3dDrawZonesがFalseの時にのみ有効です。

Xrt3dTextGetPropString()

現在のリソース値をストリングとして取得します。

```
Boolean
Xrt3dTextGetPropString(
      Widget  hText,
      int resource,
      char**str /* Returned */
      )
```

*hText*はテキスト・オブジェクトのハンドル、*resource*は(P205のリストにある、クロス・プラットフォーム名を使用した)リソースを指します。*str*は、リソース値を表す、返されたストリングのポインタです。*str*は、使用後、*Xrt3dFreePropString()*を呼び出して解放してください。

Xrt3dTextGetValues()

テキスト・オブジェクトのリソースの現在の値を取得します。

```
void
Xrt3dTextGetValues(
  Widget  text,
  ...,
  NULL
  )
```

*hText*は、テキスト・オブジェクトのハンドルです。...とある場所には、*resource-value*のペアが来ます。*resource*は、(P205のリストにある、Xrt3dPropertyクロス・プラットフォーム名として指定された)リソースを指します。*value*は、返されたリソース値を持つ変数のポインタです。*value*のデータ型は、リソースのデータ型と同じでなければなりません。このリストは、NULLで終了します。

Xrt3dTextSetPropString()

リソースをストリングで表される値にセットします。

```
Boolean
Xrt3dTextSetPropString(
      Widget      hText,
      Xrt3dProperty   resource,
      char    *str
      )
```

*hText*はテキスト・オブジェクトのハンドル、*resource*は(P205のリストにある、クロス・プラットフォーム名を使用した)リソースを指します。*str*は、リソースがセットされるストリングのポインタです。(構文はリソース・ファイルの場合と同じです。詳細はAppendix Gを参照してください。)

Xrt3dTextSetValues()

テキスト・オブジェクトのリソースをセットします。

```
void
Xrt3dTextSetValues(
  Widget text,
  ...,
  NULL
  )
```

*hText*は、テキスト・オブジェクトのハンドルです。...とある場所には、*resource-value*のペアが来ます。*resource*は、(P205のリストにある、Xrt3dPropertyクロス・プラットフォーム名として指定された)リソースを指します。*value*は、新しい値を指定します。*value*のデータ型は、リソースのデータ型と同じでなければなりません。double型のリソースにアクセスする時は、*Xrt3dDoubleToArgVal()*を使用しないでください。このリストは、NULLで終了します。

Xrt3dUnmap()

グラフ座標からpixel座標にアンマップします。

```
void
Xrt3dUnmap(
  Widget       graph,
  double       x, y, z,
  Xrt3dMapResult   *map
  )
```

アプリケーションによって使用され、*graph*をハンドルとするグラフで、座標(*x, y, z*)に対応するpixel座標を決定します。結果は、*map*に指されている構造体の*pix_x*と*pix_y*に返されます。

範囲外のアンマップのリクエストには、-1が*pix_x*と*pix_y*に返されます。

Xrt3dUnpick()

サーフェス・グリッドまたはスキャッター・プロットで、ポイントを与えられたpixel座標を決定します。

```
void
Xrt3dUnpick(
  Widget       graph,
  int      xindex, yindex,
  Xrt3dPickResult  *pick
  )
```

アプリケーションによって使用され、(*graph*をハンドルとするグラフで)データ値が表示されているpixel座標を決定します。グリッド・データでは、*xindex*と*yindex*は、アンピックされるグリッド座標のインデックスです。ポイント・データでは、*xindex*はシリーズ、*yindex*はアンピックされるポイントの番号を参照します。

結果は、*pick*で指されている構造体の*pix_x*と*pix_y*に返されます。データが範囲外の場合は、*pix_x*と*pix_y*は両方とも-1になります。

Xrt3dZInterpolate()

サーフェスにあるポイントのZ値を決定します。ポイント・データには適用されません。

```
double
Xrt3dZInterpolate(
 Widget  graph,
 double  x, y
 )
```

この手続きは、双一次補間(bilinear interpolation)を使用して計算された、(*x, y*)のサーフェス値の推定を返します。*graph*は、グラフのハンドルです。XRT3D_SURFACE_DATAがXRT3D_DATAPOINT型のデータ・セットを指している場合は、hole値が返ります。

XtCreateXrt3d()

XRT/3dウィジェットを作成します。

```
Widget
XtCreateXrt3d(
 Widget  parent,
 char*name,
 Arg *arglist,
 int argcount
 )
```

*parent*は親です。*name*は、作成されたウィジェットの名前です。*arglist*は、*argcount*引数のリストです。(*XtSetArg()*で作成)

XtCreateXrt3dText()

XRT/3dテキスト・オブジェクトを作成します。

```
Widget
XtCreateXrt3dText(
 Widget  parent,
 char*name,
 Arg *arglist,
 int argcount
 )
```

*parent*は親のウィジェット(xtXrt3dWidgetClass)です。*name*は作成されたウィジェットの名前です。*arglist*は、*argcount*引数のリストです。(*XtSetArg()*で作成)

XtIsXrt3d()

ウィジェットがXRT/3dウィジェットのサブクラスであればTrueを返します。

```
Boolean
XtIsXrt3d(
 Widget  graph
 )
```

XtIsXrt3dText()

ウィジェットがXRT/3dテキスト・オブジェクトのサブクラスであればTrueを返します。

```
Boolean
XtIsXrt3dText(
 Widget  graph
 )
```

D

トランスレーションとアクション

デフォルトのトランスレーション

アクション･ルーチン

D.1 デフォルトのトランスレーション

XRT/3dは、トランスレーション(アクションとイベントの定義)をXmPrimitiveから継承し、以下のような独自のトランスレーションを定義をします。

Event	Action
Shift <Btn2Down>	ModifyStart(Translate) DrawPreview()
Ctrl <Btn2Down>	ModifyStart(Scale) DrawPreview()
~Shift ~Ctrl <Btn2Down>	ModifyStart(Rotate) DrawPreview()
Ctrl <Btn1Down>	ModifyStart(Zoom)
<Key> r	ModifyStart(Reset) ViewReset() ModifyEnd(Reset)
<Key> x	SelectAxis(X) DrawPreview()
<Key> y	SelectAxis(Y) DrawPreview()
<Key> z	SelectAxis(Z) DrawPreview()
<Key> e	SelectAxis(Eye) DrawPreview()
<Key> n	SelectAxis(None) DrawPreview()
Shift <Btn2Motion>	TranslateUpdate() DrawPreview()
Ctrl <Btn2Motion>	ScaleUpdate() DrawPreview()
~Shift ~Ctrl <Btn2Motion>	RotateUpdate() DrawPreview()
Ctrl <Btn1Motion>	ZoomUpdate()

Event	Action
Shift <Btn2Up>	ModifyEnd(Translate)
Ctrl <Btn2Up>	ModifyEnd(Scale)
~Shift ~Ctrl <Btn2Up>	ModifyEnd(Rotate)
Ctrl <Btn1Up>	ZoomEnd() ModifyEnd(Zoom)
Button1 <Key> osfCancel*	ZoomCancel() ModifyEnd(Zoom)
Button2 <Key> osfCancel*	ModifyCancel() ModifyEnd()
Ctrl <KeyUp>	ZoomCancel()

* XRT/3dがFormまたはBulletinBoardウィジェットの内部で作成された場合、XRT/3dは、osfCancelイベントを受け取りません。キャンセルのアクションには、他のイベントを定義してください。

D.2 アクション・ルーチン

XRT/3dは、次のアクション・ルーチンを定義します。

DrawGraph()

ユーザがウィジェットとインタラクションを行っている時に、ユニット・キューブ全体を描画し、ビジュアルなフィードバックを提供します。デフォルトでは、このアクションにはイベントが定義されていません。このルーチンは、*DrawPreview()*の代わりに使用することができます。

DrawPreview()

ユーザがウィジェットとインタラクションを行っている時に、ユニット・キューブの輪郭を描画し、ビジュアルなフィードバックを提供します。インタラクティブな移動や回転が行われた時に、リアルタイムなビジュアル・フィードバックを提供するデフォルトのアクションです。

Map()

ユーザ・イベントのpixel座標にマップするグラフ座標を決定します。デフォルトでは、このアクション・ルーチンにはどのイベントも定義されていません。*Map()*は、*Xrt3dMap()*を呼び、`XtNxrt3dMapCallback`で指定されているコールバックを呼び出します。各コールバックには、グラフ座標を持つ`Xrt3dMapCallbackStruct`構造体のポインタが渡されます。

ModifyCancel()

進行中のスケーリング、移動、回転等をキャンセルし、リソースを*ModifyStart()*アクションが呼び出される前の値に戻します。

ModifyStart()

ユーザ・インタラクションを開始し、アプリケーションにそれを不許可にする方法を提供します。*ModifyStart()*は、`XtNxrt3dModifyCallback`で指定されているコールバックを呼び出し、*reason*が`XRT3D_REASON_MODIFY_START`にセットされている

Xrt3dModifyCallbackStruct構造体のポインタを各コールバックに渡します。アプリケーションは、*doit*をFalseにセットして、全てのユーザ・インタラクションを不可にすることができます。*ModifyStart()*は、行われようとするユーザ・インタラクションを示す一つのパラメータを取ります。

Translate	ユーザはウィジェットを移動しようとしている。
Scale	ユーザはウィジェットをスケーリングしようとしている。
Rotate	ユーザはウィジェットを回転しようとしている。
Zoom	ユーザはウィジェットをズームしようとしている。
Reset	ユーザはビューをリセットしようとしている。

ModifyEnd()

XRT/3dウィジェットを再描画します(XtNxrt3dxrtExposeCallbackを呼び出します)。*ModifyEnd()*は、XtNxrt3dModifyCallbackで指定されたコールバックを呼び出し、*reason* が XRT3D_REASON_MODIFY_END にセットされたXrt3dModifyCallbackStruct構造体のポインタを各コールバックに渡します。*ModifyEnd()*は、終了したユーザ・インタラクションを示すパラメータを一つ取ります。

Translate	ユーザはウィジェットの移動を終了した。
Scale	ユーザはウィジェットのスケーリングを終了した。
Rotate	ユーザはウィジェットの回転を終了した。
Zoom	ユーザはウィジェットのズームを終了した。
Reset	ユーザはビューのリセットを終了した。

NoAction()

何も行われず、コールバックも呼び出されません。デフォルトでは、このアクション・ルーチンには、どのイベントも定義されていません。このルーチンを呼ぶトランスレーションをセットして、ユーザ・イベントを不可にすることができます。例えば、スケーリングをオフにするためには、*NoAction()*を呼び出すCtrl<Btn2Motion>のトランスレーションを設定します。

Pick()

ユーザ・イベントのpixel座標に一番近いデータ値を決定します。デフォルトでは、このアクション・ルーチンには、どのイベントも定義されていません。*Pick()*は、*Xrt3dPick()*を呼び、XtNxrt3dPickCallbackで指定されているコールバックを呼び出します。各コールバックには、グラフのデータ値を持つXrt3dPickCallbackStruct構造体のポインタが渡されます。

RotateUpdate()

3Dグラフを回転します。*RotateUpdate()*は、XtNxrt3d[XYZ]Rotationリソースの新しい値を計算し、XtNxrt3dRotateCallbackで指定されているコールバックを呼び出して、Xrt3dRotateCallbackStruct構造体のポインタを各コールバックに渡します。

アプリケーションがXrt3dRotateCallbackStructの*doit*パラメータをFalseにセットしていなければ、*RotateUpdate()*は、グラフの回転の新しい値をウィジェットに適用します。

ScaleUpdate()

グラフのスケーリングを行います。*ScaleUpdate()*は、XtNxrt3dViewScaleリソースの新しい値を計算し、XtNxrt3dTransformCallbackで指定されているコールバックを呼び出して、Xrt3dTransformCallbackStruct構造体のポインタを各コールバックに渡します。アプリケーションがXrt3dTransformCallbackStructの*doit*パラメータをFalseにセットしていなければ、*ScaleUpdate()*は、新しいスケーリングの値をウィジェットに適用します。

SelectAxis()

指定した軸に沿って回転が行われるようにします。このアクションは、軸を指定する一つのパラメータを取ります。

X	X軸に沿って回転を行う。
Y	Y軸に沿って回転を行う。
Z	Z軸に沿って回転を行う。
Eye	“eye”の軸(画面に表示されている面に垂直な、キューブの中心を通る軸)に沿って回転を行う。
None	軸を特に指定しない。

TranslateUpdate()

グラフを移動します。*TranslateUpdate()*は、XtNxrt3dView[XY]Translateリソースの新しい値を計算し、XtNxrt3dTransformCallbackで指定されているコールバックを呼び出して、Xrt3dTransformCallbackStruct構造体のポインタを各コールバックに渡します。アプリケーションが、Xrt3dTransformCallbackStructの*doit*パラメータをFalseにセットしていなければ、*TranslateUpdate()*は、新しいリソース値をウィジェットに適用します。

ViewReset()

スケーリングやトランスレーションをXRT/3dのデフォルトに戻します。*ViewReset()*は、XtNxrt3dTransformCallbackで指定されているコールバックを呼び出し、XtNxrt3dViewScaleとXtNxrt3dView[XY]Translateリソースの値を持つXrt3dTransformCallbackStruct構造体のポインタを各コールバックに渡します。アプリケーションがXrt3dTransformCallbackStructの*doit*パラメータをFalseにセットしていなければ、*ViewReset()*は、新しい値をウィジェットに適用します。

ZoomUpdate()

“zoom rubberband”をアップデートします。

ZoomEnd()

“zoom rubberband” で選択されている領域をズームします。*ZoomEnd()* は、`XtNxrt3dViewScale`と`XtNxrt3dView[XY]Translate`リソースの新しい値を計算し、`XtNxrt3dTransformCallback`で指定されているコールバックを呼び出して、`Xrt3dTransformCallbackStruct`構造体のポインタを各コールバックに渡します。アプリケーションが`Xrt3dTransformCallbackStruct`の*doit*パラメータを`False`にセットしていなければ、*ZoomEnd()*は、新しいリソース値をウィジェットに適用します。

ZoomCancel()

進行中のズームをキャンセルし、“zoom rubberband”を消去します。

E

データ型

XRT/3dのデータ型のアルファベット順のリストです。構造体のCの定義も提供してあります。

Xrt3dAdjust

XtNxrt3dFooterAdjustとXtNxrt3dHeaderAdjustのリソースに使用される列挙法です。

```
XRT3D_ADJUST_LEFT
XRT3D_ADJUST_RIGHT
XRT3D_ADJUST_CENTER
```

Xrt3dAlign

XtNxrt3dLegendOrientationのリソースに使用される列挙法です。

```
XRT3D_ALIGN_VERTICAL
XRT3D_ALIGN_HORIZONTAL
```

Xrt3dAnchor

XtNxrt3dLegendAnchorのリソースに使用される列挙法です。

```
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

XtNxrt3d[XYZ]AnnoMethodの指定に使用される列挙法です。

```
XRT3D_ANNO_VALUES
XRT3D_ANNO_DATA_LABELS
XRT3D_ANNO_VALUE_LABELS
```

Xrt3dAxis

軸の指定に使用される列挙法です。

```
XRT3D_AXIS_X
XRT3D_AXIS_Y
XRT3D_AXIS_Z
XRT3D_AXIS_NONE
XRT3D_AXIS_EYE
```

Xrt3dBarFormat

XtNxrt3d[XY]BarFormatの指定に使用される列挙法です。

```
XRT3D_BAR_FIXED
XRT3D_BAR_HISTOGRAM
```

Xrt3dBorder

XtNxrt3dHeaderBorder、XtNxrt3dFooterBorder、 XtNxrt3dLegendBorder、XtNxrt3dGraphBorderとテキスト・オブジェクトのXtNxrt3dTextBorderのリソースに使用される列挙法です。

```
XRT3D_BORDER_NONE
XRT3D_BORDER_PLAIN
XRT3D_BORDER_3D_IN
XRT3D_BORDER_3D_OUT
XRT3D_BORDER_ETCHED_IN
XRT3D_BORDER_ETCHED_OUT
XRT3D_BORDER_FRAME_IN
XRT3D_BORDER_FRAME_OUT
XRT3D_BORDER_BEVEL
XRT3D_BORDER_SHADOW
```

Xrt3dCallbackStruct

XtNxrt3dResizeCallbackとXtNxrt3dExposeCallbackのリソースで定義されたコールバックに使用される構造体です。これらのリソースの詳細は、Appendix Aのリソースの解説を参照してください。

```
typedef struct {
 int reason;
 XEvent  *event;
 Window  window;
 Region  region;
 int change;
} Xrt3dCallbackStruct
```

Xrt3dColor

*char**と定義される、プラットフォーム固有の型です。

Xrt3dContourStyle

等高線とゾーンの表示を定義する構造体です。スキャッター・プロットには適用されません。

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

  unsigned long    res1; /* reserved */
  unsigned long    res2; /* reserved */
} Xrt3dContourStyle;
```

Xrt3dData

XtNxrt3dSurfaceDataとXtNxrt3dZoneDataのデータを定義する構造体です。二つのデータ構造体の共用体(union)です。

```
typedef union {
  Xrt3dGridData    g;
  Xrt3dIrGridData  ig;
  Xrt3dPointData   p;
} Xrt3dData;
```

Xrt3dDataStyle

スキャッター・プロットの各シリーズに対する線とシンボルのスタイルの定義に使用される構造体です。

```
typedef struct {
      Xrt3dLineStyle       line_style;
      Xrt3dSymbolStyle     symbol_style;
} Xrt3dDataStyle;
```

Xrt3dDataType

Xrt3dData構造体の型の定義に使用される列挙法です。

```
XRT3D_DATA_GRID = 1
XRT3D_DATA_IRGRID
XRT3D_DATA_POINT
```

Xrt3dDistnMethod

DrawContoursと(または)DrawZonesがTrueの場合に、分布法(distribution method)を定義する列挙法です。二つの方法があります。スキャッター・プロットには適用されません。

```
XRT3D_DISTN_LINEAR
XRT3D_DISTN_FROM_TABLE
```

Xrt3dDistnRange

分布表(distribution table)の分布の範囲の定義に使用される列挙法です。二つの方法があります。スキャッター・プロットには適用されません。

```
XRT3D_DISTN_RANGE_DATA
XRT3D_DISTN_RANGE_ALL
```

Xrt3dDistnTable

任意の分布表(distribution table)を定義する構造体です。スキャッター・プロットには適用されません。

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

Xrt3dDrawPSColor

*Xrt3dDrawPS()*で色の処理を指定する列挙法です。

```
XRT3D_PS_COLOR_AS_IS
XRT3D_PS_COLOR_AUTO
XRT3D_PS_COLOR_MONO
```

Xrt3dGridData

型がXRT3D_DATA_GRIDの時に、Xrt3dDataを定義する構造体です。

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

Xrt3dImage

XtPointerと定義される、プラット・フォーム固有の型です。

この型は、同様にXtPointerとして定義されるXrtPDSImageと一緒に使用できます。

Xrt3dImageLayout

XtNxrt3dImageLayout、XtNxrt3dGraphImageLayout、XtNxrt3dHeaderImageLayout、XtNxrt3dFooterImageLayout、XtNxrt3dLegendImageLayout及び、テキスト・オブジェクトのXtNxrt3dTextImageLayoutに使用される列挙法です。

```
XRT3D_IMAGELAYOUT_CENTERED = 1
XRT3D_IMAGELAYOUT_TILED
```

Xrt3dInterpMethod

*Xrt3dDataWindow()*で、補間の方法の定義に使用される列挙法です。スキャッター・プロットには適用されません。

```
XRT3D_INTERP_LINEAR_SPLINE
XRT3D_INTERP_CUBIC_SPLINE
```

Xrt3dIrGridData

型がXRT3D_DATA_IRGRIDの時に、Xrt3dDataを定義する構造体です。

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

Xrt3dLegendStyle

XtNxrt3dLegendStyleのリソースで使用される凡例スタイルの列挙法です。スキャッター・プロットには適用されません。

```
XRT3D_LEGEND_STYLE_STEPPED
XRT3D_LEGEND_STYLE_CONTINUOUS
```

Xrt3dLinePattern

Xrt3dContourStyle構造体内で使用される線のパターンの列挙法です。等高線、グリッド線、スキャッター・プロットのドロップ・ラインに適用されます。

```
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
```

Xrt3dLineStyle

線のパターン、色、幅を定義する構造体です。グリッド線、スキャッター・プロットのドロップ・ラインに適用されます。

```
typedef struct {
      Xrt3dLinePattern     pattern;
      Xrt3dColor       color;
      int          width;
} Xrt3dLineStyle
```

Xrt3dMapCallbackStruct

XtNxrt3dMapCallbackリストのコールバックに渡される情報を定義する構造体です。

```
typedef struct {
  int         reason;      // Read-only
  XEvent      *event;      // Read-only
  String      *params;     // Read-only
  Cardinal        nparams;     // Read-only
  Xrt3dRegion     region;      // Read-only
  Xrt3dMapResult  map;         // Read-only
} Xrt3dMapCallbackStruct;
```

Xrt3dMapResult

マップされたpixel座標に関する情報を渡す構造体です。

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

Xrt3dModifyCallbackStruct

XtNxrt3dModifyCallbackリストのコールバックに渡される情報を定義する構造体です。

```
typedef struct {
 int     reason; // Read-only
 XEvent  *event; // Read-only
 String  *params; // Read-only
 Cardinal    nparams; // Read-only
 Boolean doit;
} Xrt3dModifyCallbackStruct;
```

Xrt3dPickCallbackStruct

XtNxrt3dPickCallbackリストのコールバックに渡される情報を定義する構造体です。

```
typedef struct {
 int         reason; // Read-only
 XEvent      *event; // Read-only
 String      *params; // Read-only
 Cardinal        nparams; // Read-only
 Xrt3dRegion     region; // Read-only
 Xrt3dPickResult pick; // Read-only
} Xrt3dPickCallbackStruct;
```

Xrt3dPickResult

ピックされたpixel座標に関する情報を渡す構造体です。

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

Xrt3dPoint3D

スキャッター・プロットのポイント構造体です。

```
typedef struct {
     double      x, y, z;
     }
```

Xrt3dPointData

XRT3D_DATA_POINTの場合のXrt3dDataを定義する構造体です。

```
typedef struct {
     Xrt3dDataType   type;
     int     nseries;
     double      noval;
     Xrt3dPointSeries *series;
} Xrt3dPointData;
```

Xrt3dPointSeries

スキャッター・プロットにおけるシリーズ内のポイントのリストを持つ構造体です。

```
typedef struct {
     int     npoints;
     Xrt3dPoint3D *npoints;
} Xrt3dPointSeries;
```

Xrt3dPreviewMethod

インタラクティブな回転が行われている時の表示法を指定する列挙法です。

```
XRT3D_PREVIEW_CUBE
XRT3D_PREVIEW_FULL
```

Xrt3dProperty

Xrt3dGetPropString() 、 *Xrt3dGetValues()* 、 *Xrt3dSetPropString()* 、 *Xrt3dSetValues()* 、 *Xrt3dTextGetPropString()*、*Xrt3dTextGetValues()*、*Xrt3dTextSetValues()*、*Xrt3dTextSetPropString()* の手続きに使用される、リソースのクロス・プラットフォーム名の列挙法です。各リソースの定義のリストは、P205にあります。

Xrt3dReason

コールバックの理由(reason)の定義に使用される列挙法です。XtNxrt3dMapCallback、XtNxrt3dModifyCallback、XtNxrt3dPickCallback、XtNxrt3dRotateCallback、XtNxrt3dTransformCallback構造体に使用されます。

```
XRT3D_REASON_MODIFY_START
XRT3D_REASON_MODIFY_END
XRT3D_REASON_ROTATE
XRT3D_REASON_TRANSFORM
XRT3D_REASON_MAP
XRT3D_REASON_PICK
```

Xrt3dRegion

マップまたはピックの結果の列挙法です。*Xrt3dMap()*と*Xrt3dPick()*によって返されます。

```
XRT3D_RGN_NOWHERE
XRT3D_RGN_IN_GRAPH
XRT3D_RGN_IN_LEGEND
XRT3D_RGN_IN_FOOTER
XRT3D_RGN_IN_HEADER
```

Xrt3dRotateCallbackStruct

XtNxrt3dRotateCallbackリストのコールバックに渡される情報を定義する構造体です。

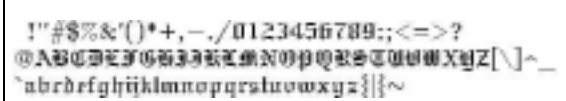
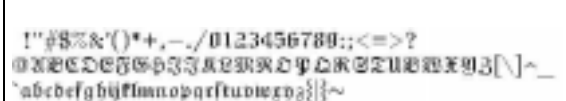
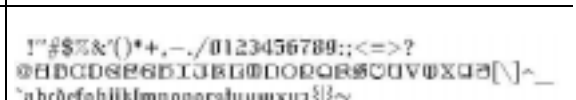
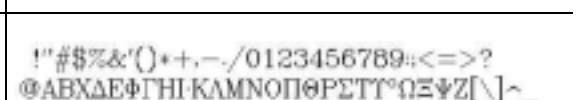
```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  String  *params; // Read-only
  Cardinal    nparams; // Read-only
  double  xrotation;
  double  yrotation;
  double  zrotation;
  Boolean doit;
```

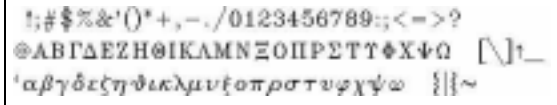
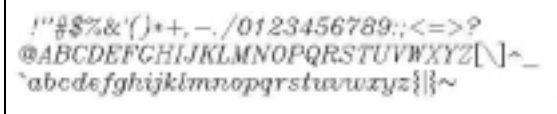
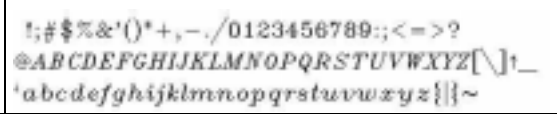
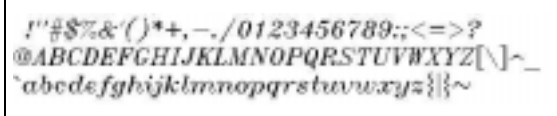
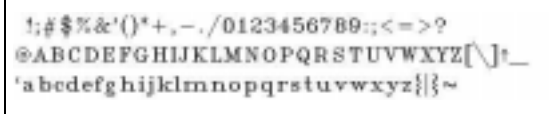
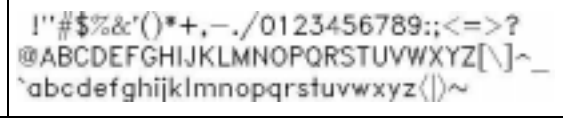
```
} Xrt3dRotateCallbackStruct;
```

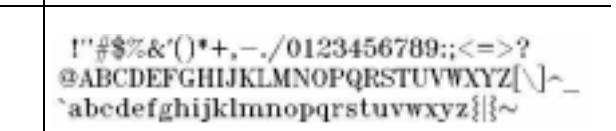
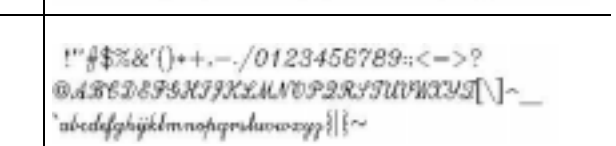
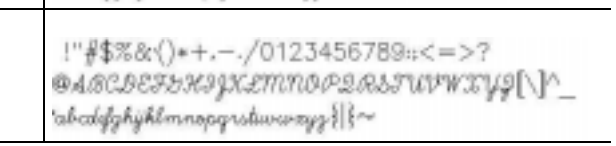
Xrt3dStrokeFont

stroke フォント名の列挙法です。XtNxrt3dAxisStrokeFont と XtNxrt3dAxisTitleStrokeFontに使用されます。

```
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	

Stroke Font Name	Glyphs
Greek Complex Small	
Greek Simplex	
Italic Complex	
Italic Complex Small	
Italic Triplex	
Roman Complex	
Roman Complex Small	
Roman Duplex	

Stroke Font Name	Glyphs
Roman Simplex	
Roman Triplex	
Script Complex	
Script Simplex	

Xrt3dSymbolPattern

スキャッター・プロットのポイントに使用されるシンボルのパターンの列挙法です。

```
XRT3D_SYMBOL_NONE
XRT3D_SYMBOL_DOT
XRT3D_SYMBOL_BOX
XRT3D_SYMBOL_TRI
XRT3D_SYMBOL_DIAMOND
XRT3D_SYMBOL_STAR
XRT3D_SYMBOL_VERT_LINE
XRT3D_SYMBOL_HORIZ_LINE
XRT3D_SYMBOL_CROSS
XRT3D_SYMBOL_CIRCLE
XRT3D_SYMBOL_SQUARE
XRT3D_SYMBOL_INVERT_TRI
XRT3D_SYMBOL_DIAG_CROSS
XRT3D_SYMBOL_OPEN_TRI
XRT3D_SYMBOL_OPEN_DIAMOND
XRT3D_SYMBOL_OPEN_INVERT_TRI
```

Xrt3dSymbolStyle

スキャッター・プロットにおいて、指定されたデータ・シリーズに対するシンボルの色、サイズ、パターンを定義する構造体です。

```
typedef struct {
     Xrt3dSymbolPattern   pattern;
     Xrt3dColor       color;
     int          size;
} Xrt3dSymbolStyle
```

Xrt3dTextAttachMethod

どのようにXRT/3dのテキスト・オブジェクトがグラフに貼り付けられるかを指定する列挙法です。

```
XRT3D_TEXT_ATTACH_INDEX
XRT3D_TEXT_ATTACH_PIXEL
XRT3D_TEXT_ATTACH_POINT
```

Xrt3dTransformCallbackStruct

XtNxrt3dTransformCallbackリストのコールバックに渡される情報を定義する構造体です。

```
typedef struct {
  int     reason; // Read-only
  XEvent  *event; // Read-only
  String  *params; // Read-only
  Cardinal    nparams; // Read-only
  double  scale;
  double  xtranslate;
  double  ytranslate;
  Boolean doit;
} Xrt3dTransformCallbackStruct;
```

Xrt3dType

XtNxrt3dTypeの指定に使用される列挙法です。

```
XRT3D_TYPE_SURFACE
XRT3D_TYPE_BAR
XRT3D_TYPE_SCATTER
```

Xrt3dValueLabel

軸の値ラベルを提供する構造体です。

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

Xrt3dXYColor

各棒の色の指定に使用される構造体です。スキャッター・プロットには適用されません。

```
typedef struct {
  int xindex;
  int yindex;
  char*color;
  unsigned long    res; /* reserved */
} Xrt3dXYColor;
```

Xrt3dZoneMethod

XtNxrt3dZoneMethodに使用され、ゾーンの塗りつぶし法を指定する列挙法です。スキャッター・プロットには適用されません。

```
XRT3D_ZONE_CONTOURS
XRT3D_ZONE_CELLS
```

F

リソース名、型とクラス

リソースの型とクラスのアルファベット順のリストです。各リソース名の下には、Xrt3dPropertyの列挙法が記載されています。

Name	Class	Type
XtNxrt3dAxisStrokeFont XRT3D_AXIS_STROKE_FONT	XtCXrt3dStrokeFont	XtRXrt3dStrokeFont
XtNxrt3dAxisStrokeSize XRT3D_AXIS_STROKE_SIZE	XtCXrt3dStrokeFontSize	XtRDimension
XtNxrt3dAxisTitleStrokeFont XRT3D_AXIS_TITLE_STROKE_FONT	XtCXrt3dStrokeFont	XtRXrt3dStrokeFont
XtNxrt3dAxisTitleStrokeSize XRT3D_AXIS_TITLE_STROKE_SIZE	XtCXrt3dStrokeFontSize	XtRDimension
XtNxrt3dBackgroundColor XRT3D_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtRXrt3dColor
XtNxrt3dColorFunc XRT3D_COLOR_FUNC	XtCXrt3dColorFunc	XtRFunction
XtNxrt3dContourStyles XRT3D_CONTOUR_STYLES	XtCXrt3dContourStyles	XtRXrt3dContourStyles
XtNxrt3dDataAreaBackgroundColor XRT3D_DATA_AREA_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtR3dColor
XtNxrt3dDataStyles XRT3D_DATA_STYLES	XtCXrt3dDataStyles	XtRXrt3dDataStyles
XtNxrt3dDataStylesUseDefault XRT3D_DATA_STYLES_USE_DEFAULT	XtCXrt3dDataStylesUseDefault	XtRBoolean
XtNxrt3dDebug XRT3D_DEBUG	XtCXrt3dDebug	XtRBoolean
XtNxrt3dDistnMethod XRT3D_DISTN_METHOD	XtCXrt3dDistnMethod	XtRXrt3dDistnMethod

Name	Class	Type
XtNxrt3dDistnTable XRT3D_DISTN_TABLE	XtCXrt3dDistnTable	XtRXrt3dDistnTable
XtNxrt3dDoubleBuffer XRT3D_DOUBLE_BUFFER	XtCXrt3dDoubleBuffer	XtRBoolean
XtNxrt3dDrawContours XRT3D_DRAW_CONTOURS	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dDrawDropLines XRT3D_DRAW_DROPLINES	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dDrawHiddenLines XRT3D_DRAW_HIDDEN_LINES	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dDrawMesh XRT3D_DRAW_MESH	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dDrawShaded XRT3D_DRAW_SHADED	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dDrawZones XRT3D_DRAW_ZONES	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dExposeCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dFooterAdjust XRT3D_FOOTER_ADJUST	XtCXrt3dAdjust	XtRXrt3dAdjust
XtNxrt3dFooterBackgroundColor XRT3D_FOOTER_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtRXrt3dColor
XtNxrt3dFooterBorder XRT3D_FOOTER_BORDER	XtCXrt3dBorder	XtRXrt3dBorder
XtNxrt3dFooterBorderWidth XRT3D_FOOTER_BORDER_WIDTH	XtCXrt3dBorderWidth	XtRDimension
XtNxrt3dFooterFont XRT3D_FOOTER_FONT	XmCFontList	XmRFontList
XtNxrt3dFooterForegroundColor XRT3D_FOOTER_FOREGROUND_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dFooterHeight XRT3D_FOOTER_HEIGHT	XtCXrt3dDimension	XtRDimension
XtNxrt3dFooterImage XRT3D_FOOTER_IMAGE	XtCXrt3dImage	XtRXrt3dImage
XtNxrt3dFooterImageLayout XRT3D_FOOTER_IMAGE_LAYOUT	XtCXrt3dImageLayout	XtRXrt3dImageLayout
XtNxrt3dFooterImageMinimumSize XRT3D_FOOTER_IMAGE_MINIMUM_SIZE	XtCXrt3dImageMinimumSize	XtRBoolean
XtNxrt3dFooterStrings XRT3D_FOOTER_STRINGS	XtCXrt3dStrings	XtRXrt3dStrings
XtNxrt3dFooterWidth XRT3D_FOOTER_WIDTH	XtCXrt3dDimension	XtRDimension

Name	Class	Type
XtNxrt3dFooterX XRT3D_FOOTER_X	XtCXrt3dPosition	XtRPosition
XtNxrt3dFooterXUseDefault XRT3D_FOOTER_X_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dFooterY XRT3D_FOOTER_Y	XtCXrt3dPosition	XtRPosition
XtNxrt3dFooterYUseDefault XRT3D_FOOTER_Y_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dForegroundColor XRT3D_FOREGROUND_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dGraphBackgroundColor XRT3D_GRAPH_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtRXrt3dColor
XtNxrt3dGraphBorder XRT3D_GRAPH_BORDER	XtCXrt3dBorder	XtRXrt3dBorder
XtNxrt3dGraphBorderWidth XRT3D_GRAPH_BORDER_WIDTH	XtCXrt3dBorderWidth	XtRDimension
XtNxrt3dGraphForegroundColor XRT3D_GRAPH_FOREGROUND_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dGraphHeight XRT3D_GRAPH_HEIGHT	XtCXrt3dDimension	XtRDimension
XtNxrt3dGraphHeightUseDefault XRT3D_GRAPH_HEIGHT_USE_DEFAULT	XtCXrt3dDimensionUseDefault	XtRBoolean
XtNxrt3dGraphImage XRT3D_GRAPH_IMAGE	XtCXrt3dImage	XtRXrt3dImage
XtNxrt3dGraphImageLayout XRT3D_GRAPH_IMAGE_LAYOUT	XtCXrt3dImageLayout	XtRXrt3dImageLayout
XtNxrt3dGraphWidth XRT3D_GRAPH_WIDTH	XtCXrt3dDimension	XtRDimension
XtNxrt3dGraphWidthUseDefault XRT3D_GRAPH_WIDTH_USE_DEFAULT	XtCXrt3dDimensionUseDefault	XtRBoolean
XtNxrt3dGraphX XRT3D_GRAPH_X	XtCXrt3dPosition	XtRPosition
XtNxrt3dGraphXUseDefault XRT3D_GRAPH_X_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dGraphY XRT3D_GRAPH_Y	XtCXrt3dPosition	XtRPosition
XtNxrt3dGraphYUseDefault XRT3D_GRAPH_Y_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dHeaderAdjust XRT3D_HEADER_ADJUST	XtCXrt3dAdjust	XtRXrt3dAdjust

Name	Class	Type
XtNxrt3dHeaderBackgroundColor XRT3D_HEADER_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtRXrt3dColor
XtNxrt3dHeaderBorder XRT3D_HEADER_BORDER	XtCXrt3dBorder	XtRXrt3dBorder
XtNxrt3dHeaderBorderWidth XRT3D_HEADER_BORDER_WIDTH	XtCXrt3dBorderWidth	XtRDimension
XtNxrt3dHeaderFont XRT3D_HEADER_FONT	XmCFontList	XmRFontList
XtNxrt3dHeaderForegroundColor XRT3D_HEADER_FOREGROUND_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dHeaderHeight XRT3D_HEADER_HEIGHT	XtCXrt3dDimension	XtRDimension
XtNxrt3dHeaderImage XRT3D_HEADER_IMAGE	XtCXrt3dImage	XtRXrt3dImage
XtNxrt3dHeaderImageLayout XRT3D_HEADER_IMAGE_LAYOUT	XtCXrt3dImageLayout	XtRXrt3dImageLayout
XtNxrt3dHeaderImageMinimumSize XRT3D_HEADER_IMAGE_MINIMUM_SIZE	XtCXrt3dImageMinimumSize	XtRBoolean
XtNxrt3dHeaderStrings XRT3D_HEADER_STRINGS	XtCXrt3dStrings	XtRXrt3dStrings
XtNxrt3dHeaderWidth XRT3D_HEADER_WIDTH	XtCXrt3dDimension	XtRDimension
XtNxrt3dHeaderX XRT3D_HEADER_X	XtCXrt3dPosition	XtRPosition
XtNxrt3dHeaderXUseDefault XRT3D_HEADER_X_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dHeaderY XRT3D_HEADER_Y	XtCXrt3dPosition	XtRPosition
XtNxrtHeaderYUseDefault XRT3D_HEADER_Y_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dImage XRT3D_IMAGE	XtCXrt3dImage	XtRXrt3dImage
XtNxrt3dImageLayout XRT3D_IMAGE_LAYOUT	XtCXrt3dImageLayout	XtRXrt3dImageLayout
XtNxrt3dLegendAnchor XRT3D_LEGEND_ANCHOR	XtCXrt3dAnchor	XtRXrt3dAnchor
XtNxrt3dLegendBackgroundColor XRT3D_LEGEND_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtRXrt3dColor
XtNxrt3dLegendBorder XRT3D_LEGEND_BORDER	XtCXrt3dBorder	XtRXrt3dBorder

Name	Class	Type
XtNxrt3dLegendBorderWidth XRT3D_LEGEND_BORDER_WIDTH	XtCXrt3dBorderWidth	XtRDimension
XtNxrt3dLegendDistnRange XRT3D_LEGEND_DISTN_RANGE	XtCXrt3dLegendDistnRange	XtRXrt3dDistnRange
XtNxrt3dLegendFont XRT3D_LEGEND_FONT	XmCFontList	XmRFontList
XtNxrt3dLegendForegroundColor XRT3D_LEGEND_FOREGROUND_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dLegendHeight XRT3D_LEGEND_HEIGHT	XtCXrt3dDimension	XtRDimension
XtNxrt3dLegendImage XRT3D_LEGEND_IMAGE	XtCXrt3dImage	XtRXrt3dImage
XtNxrt3dLegendImageLayout XRT3D_LEGEND_IMAGE_LAYOUT	XtCXrt3dImageLayout	XtRXrt3dImageLayout
XtNxrt3dLegendLabelFilterc XRT3D_LEGEND_LABEL_FILTER	XtCXrt3dLegendLabelFilter	XtRInt
XtNxrt3dLegendLabelFunc XRT3D_LEGEND_LABEL_FUNC	XtCXrt3dLegendLabelFunc	XtRFunction
XtNxrt3dLegendOrientation XRT3D_LEGEND_ORIENTATION	XtCXrt3dOrientation	XtRXrt3dAlign
XtNxrt3dLegendShow XRT3D_LEGEND_SHOW	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dLegendStrings XRT3D_LEGEND_STRINGS	XtCXrt3dStrings	XtRXrt3dStrings
XtNxrt3dLegendStyle XRT3D_LEGEND_STYLE	XtCXrt3dLegendStyle	XtRXrt3dLegendStyle
XtNxrt3dLegendTitle XRT3D_LEGEND_TITLE	XtCXrt3dLegendTitle	XtRXrt3dString
XtNxrt3dLegendWidth XRT3D_LEGEND_WIDTH	XtCXrt3dDimension	XtRDimension
XtNxrt3dLegendX XRT3D_LEGEND_X	XtCXrt3dPosition	XtRPosition
XtNxrt3dLegendXUseDefault XRT3D_LEGEND_X_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dLegendY XRT3D_LEGEND_Y	XtCXrt3dPosition	XtRPosition
XtNxrt3dLegendYUseDefault XRT3D_LEGEND_Y_USE_DEFAULT	XtCXrt3dPositionUseDefault	XtRBoolean
XtNxrt3dMapCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dMeshBottomColor XRT3D_MESH_BOTTOM_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor

Name	Class	Type
XtNxrt3dMeshTopColor XRT3D_MESH_TOP_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dModifyCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dNumDistnLevels XRT3D_NUM_DISTN_LEVELS	XtCXrt3dNumDistnLevels	XtRInt
XtNxrt3dPerspectiveDepth XRT3D_PERSPECTIVE_DEPTH	XtCXrt3dPerspectiveDepth	XtRXrt3dDouble
XtNxrt3dPickCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dPreviewMethod XRT3D_PREVIEW_METHOD	XtCXrt3dPreviewMethod	XtRXrt3dPreviewMethod
XtNxrt3dProjectZMax XRT3D_PROJECT_ZMAX	XtCXrt3dProject	XtRXrt3dProject
XtNxrt3dProjectZMin XRT3D_PROJECT_ZMIN	XtCXrt3dProject	XtRXrt3dProject
XtNxrt3dRepaint XRT3D_REPAINT	XtCXrt3dRepaint	XtRBoolean
XtNxrt3dResizeCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dRotateCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dSolidSurface XRT3D_SOLID_SURFACE	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dSurfaceBottomColor XRT3D_SURFACE_BOTTOM_COLOR	XtCXrt3dForegroundColor	XtRString
XtNxrt3dSurfaceData XRT3D_SURFACE_DATA	XtCXrt3dData	XtRXrt3dData
XtNxrt3dSurfaceTopColor XRT3D_SURFACE_TOP_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dTransformCallback[a]	XtCXrt3dCallback	XtRCallback
XtNxrt3dType XRT3D_TYPE	XtCXrt3dType	XtRXrt3dType
XtNxrt3dViewNormalized XRT3D_VIEW_NORMALIZED	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dViewScale XRT3D_VIEW_SCALE	XtCXrt3dViewControl	XtRXrt3dDouble
XtNxrt3dViewXTranslate XRT3D_VIEW_XTRANSLATE	XtCXrt3dViewControl	XtRXrt3dDouble
XtNxrt3dViewYTranslate XRT3D_VIEW_YTRANSLATE	XtCXrt3dViewControl	XtRXrt3dDouble
XtNxrt3dXAnnoMethod XRT3D_XANNO_METHOD	XtCXrt3dAnnoMethod	XtRXrt3dAnnoMethod

Name	Class	Type
XtNxrt3dXAxisShow XRT3D_XAXIS_SHOW	XtCXrt3dAxisShow	XtRBoolean
XtNxrt3dXAxisTitle XRT3D_XAXIS_TITLE	XtCXrt3dAxisTitle	XtRXrt3dString
XtNxrt3dXBarFormat XRT3D_XBAR_FORMAT	XtCXrt3dBarFormat	XtRXrt3dBarFormat
XtNxrt3dXBarSpacing XRT3D_XBAR_SPACING	XtCXrt3dBarSpacing	XtRXrt3dDouble
XtNxrt3dXDataLabels XRT3D_XDATA_LABELS	XtCXrt3dDataLabels	XtRXrt3dStrings
XtNxrt3dXGridLines XRT3D_XGRID_LINES	XtCXrt3dGridLines	XtRXrt3dGridLines
XtNxrt3dXGridLineStyle XRT3D_XGRID_LINE_STYLE	XtCXrt3dGridLineStyle	XtRXrt3dLineStyle
XtNxrt3dXGridLineStyleUseDefault XRT3D_XGRID_LINE_STYLE_USE_DEFAULT	XtCXrt3dGridLineStyleUseDefault	XtRBoolean
XtNxrt3dXMax XRT3D_XMAX	XtCXrt3dAxisBounds	XtR3dDouble
XtNxrt3dXMaxUseDefault XRT3D_XMAX_USE_DEFAULT	XtCXrt3dAxisBoundsUseDefault	XtRBoolean
XtNxrt3dXMeshFilter XRT3D_XMESH_FILTER	XtCXrt3dMeshFilter	XtRInt
XtNxrt3dXMeshShow XRT3D_XMESH_SHOW	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dXMin XRT3D_XMIN	XtCXrt3dAxisBounds	XtRXrt3dDouble
XtNxrt3dXMinUseDefault XRT3D_XMIN_USE_DEFAULT	XtCXrt3dAxisBoundsUseDefault	XtRBoolean
XtNxrt3dXRotation XRT3D_XROTATION	XtCXrt3dRotation	XtRXrt3dDouble
XtNxrt3dXScale XRT3D_XSCALE	XtCXrt3dScale	XtRXrt3dDouble
XtNxrt3dXValueLabels XRT3D_XVALUE_LABELS	XtCXrt3dValueLabels	XtRXrt3dValueLabels
XtNxrt3dXYColors XRT3D_XY_COLORS	XtCXrt3dXYColors	XtRXrt3dXYColors
XtNxrt3dYAnnoMethod XRT3D_YANNO_METHOD	XtCXrt3dAnnoMethod	XtRXrt3dAnnoMethod
XtNxrt3dYAxisShow XRT3D_YAXIS_SHOW	XtCXrt3dAxisShow	XtRBoolean

Name	Class	Type
XtNxrt3dYAxisTitle XRT3D_YAXIS_TITLE	XtCXrt3dAxisTitle	XtRXrt3dString
XtNxrt3dYBarFormat XRT3D_YBAR_FORMAT	XtCXrt3dBarFormat	XtRXrt3dBarFormat
XtNxrt3dYBarSpacing XRT3D_YBAR_SPACING	XtCXrt3dBarSpacing	XtRXrt3dDouble
XtNxrt3dYDataLabels XRT3D_YDATA_LABELS	XtCXrt3dDataLabels	XtRXrt3dStrings
XtNxrt3dYGridLines XRT3D_YGRID_LINES	XtCXrt3dGridLines	XtRXrt3dGridLines
XtNxrt3dYGridLineStyle XRT3D_YGRID_LINE_STYLE	XtCXrt3dGridLineStyle	XtRXrt3dLineStyle
XtNxrt3dYGridLineStyleUseDefault XRT3D_YGRID_LINE_STYLE_USE_DEFAULT	XtCXrt3dGridLineStyleUseDefault	XtRBoolean
XtNxrt3dYMax XRT3D_YMAX	XtCXrt3dAxisBounds	XtRXrt3dDouble
XtNxrt3dYMaxUseDefault XRT3D_YMAX_USE_DEFAULT	XtCXrt3dAxisBoundsUseDefault	XtRBoolean
XtNxrt3dYMeshFilter XRT3D_YMESH_FILTER	XtCXrt3dMeshFilter	XtRInt
XtNxrt3dYMeshShow XRT3D_YMESH_SHOW	XtCXrt3dDrawOption	XtRBoolean
XtNxrt3dYMin XRT3D_YMIN	XtCXrt3dAxisBounds	XtRXrt3dDouble
XtNxrt3dYMinUseDefault XRT3D_YMIN_USE_DEFAULT	XtCXrt3dAxisBoundsUseDefault	XtRBoolean
XtNxrt3dYRotation XRT3D_YROTATION	XtCXrt3dRotation	XtRXrt3dDouble
XtNxrt3dYScale XRT3D_YSCALE	XtCXrt3dScale	XtRXrt3dDouble
XtNxrt3dYValueLabels XRT3D_YVALUE_LABELS	XtCXrt3dValueLabels	XtRXrt3dValueLabels
XtNxrt3dZAnnoMethod XRT3D_ZANNO_METHOD	XtCXrt3dAnnoMethod	XtRXrt3dAnnoMethod
XtNxrt3dZAxisShow XRT3D_ZAXIS_SHOW	XtCXrt3dAxisShow	XtRBoolean
XtNxrt3dZAxisTitle XRT3D_ZAXIS_TITLE	XtCXrt3dAxisTitle	XtRString
XtNxrt3dZGridLines XRT3D_ZGRID_LINES	XtCXrt3dGridLines	XtRXrt3dGridLines

Name	Class	Type
XtNxrt3dZGridLineStyle XRT3D_ZGRID_LINE_STYLE	XtCXrt3dGridLineStyle	XtRXrt3dLineStyle
XtNxrt3dZGridLineStyleUseDefault XRT3D_ZGRID_LINE_STYLE_USE_DEFAULT	XtCXrt3dGridLineStyleUseDefault	XtRBoolean
XtNxrt3dZMax XRT3D_ZMAX	XtCXrt3dAxisBounds	XtRXrt3dDouble
XtNxrt3dZMaxUseDefault XRT3D_ZMAX_USE_DEFAULT	XtCXrt3dAxisBoundsUseDefault	XtRBoolean
XtNxrt3dZMin XRT3D_ZMIN	XtCXrt3dAxisBounds	XtRXrt3dDouble
XtNxrt3dZMinUseDefault XRT3D_ZMIN_USE_DEFAULT	XtCXrt3dAxisBoundsUseDefault	XtRBoolean
XtNxrt3dZoneData XRT3D_ZONE_DATA	XtCXrt3dData	XtRXrt3dData
XtNxrt3dZoneMethod XRT3D_ZONE_METHOD	XtCXrt3dZoneMethod	XtRXrt3dZoneMethod
XtNxrt3dZOrigin XRT3D_ZORIGIN	XtCXrt3dOrigin	XtRXrt3dDouble
XtNxrt3dZRotation XRT3D_ZROTATION	XtCXrt3dRotation	XtRXrt3dDouble
XtNxrt3dZScale XRT3D_ZSCALE	XtCXrt3dScale	XtRXrt3dDouble
XtNxrt3dZValueLabels XRT3D_ZVALUE_LABELS	XtCXrt3dValueLabels	XtRXrt3dValueLabels

a: このリソースは、Windowsでは使用されないのでクロス・プラットフォーム名がありません。

XRT/3d テキスト・オブジェクト リソース

Name	Class	Type
XtNxrt3dTextAdjust XRT3D_TEXT_ADJUST	XtCXrt3dAdjust	XtRXrt3dAdjust
XtNxrt3dTextAttachIndexX XRT3D_TEXT_ATTACH_INDEX_X	XtCXrt3dTextAttachIndex	XtRInt
XtNxrt3dTextAttachIndexY XRT3D_TEXT_ATTACH_INDEX_Y	XtCXrt3dTextAttachIndex	XtRInt
XtNxrt3dTextAttachIndexPoint XRT3D_TEXT_ATTACH_INDEX_POINT	XtCXrt3dTextAttachIndex	XtRInt
XtNxrt3dTextAttachIndexSeries XRT3D_TEXT_ATTACH_INDEX_SERIES	XtCXrt3dTextAttachIndex	XtRInt

XRT/3d テキスト･オブジェクト リソース

Name	Class	Type
XtNxrt3dTextAttachMethod XRT3D_TEXT_ATTACH_METHOD	XtCXrt3dTextAttachMethod	XtRXrt3dAttachMethod
XtNxrt3dTextAttachPixelX XRT3D_TEXT_ATTACH_PIXEL_X	XtCXrt3dTextAttachPixel	XtRPosition
XtNxrt3dTextAttachPixelY XRT3D_TEXT_ATTACH_PIXEL_Y	XtCXrt3dTextAttachPixel	XtRPosition
XtNxrt3dTextAttachPointX XRT3D_TEXT_ATTACH_POINT_X	XtCXrt3dTextAttachPoint	XtRXrt3dDouble
XtNxrt3dTextAttachPointY XRT3D_TEXT_ATTACH_POINT_Y	XtCXrt3dTextAttachPoint	XtRXrt3dDouble
XtNxrt3dTextAttachPointZ XRT3D_TEXT_ATTACH_POINT_Z	XtCXrt3dTextAttachPoint	XtRXrt3dDouble
XtNxrt3dTextBackgroundColor XRT3D_TEXT_BACKGROUND_COLOR	XtCXrt3dBackgroundColor	XtRXrt3dColor
XtNxrt3dTextBorder XRT3D_TEXT_BORDER	XtCXrt3dBorder	XtRXrt3dBorder
XtNxrt3dTextBorderWidth XRT3D_TEXT_BORDER_WIDTH	XtCXrt3dBorderWidth	XtRDimension
XtNxrt3dTextFont XRT3D_TEXT_FONT	XmCFontList	XmRFontList
XtNxrt3dTextForegroundColor XRT3D_TEXT_FOREGROUND_COLOR	XtCXrt3dForegroundColor	XtRXrt3dColor
XtNxrt3dTextImage XRT3D_TEXT_IMAGE	XtCXrt3dImage	XtRXrt3dImage
XtNxrt3dTextImageLayout XRT3D_TEXT_IMAGE_LAYOUT	XtCXrt3dImageLayout	XtRXrt3dImageLayout
XtNxrt3dTextImageMinimumSize XRT3D_TEXT_IMAGE_MINIMUM_SIZE	XtCXrt3dImageMinimumSize	XtRBoolean
XtNxrt3dTextLineShow XRT3D_TEXT_LINE_SHOW	XtCXrt3dTextDrawOptions	XtRBoolean
XtNxrt3dTextList XRT3D_TEXT_LIST	XtCXrt3dTextList	XtRXrt3dTextList
XtNxrt3dTextOffsetX XRT3D_TEXT_OFFSET_X	XtCXrt3dPosition	XtRPosition
XtNxrt3dTextOffsetY XRT3D_TEXT_OFFSET_Y	XtCXrt3dPosition	XtRPosition
XtNxrt3dTextPlane XRT3D_TEXT_PLANE	XtCXrt3dTextPlane	XtRXrt3dPlane
XtNxrt3dTextPrintFont XRT3D_TEXT_PRINT_FONT	XtCString	XtRXrt3dString

XRT/3d テキスト･オブジェクト リソース

Name	Class	Type
XtNxrt3dTextShow XRT3D_TEXT_SHOW	XtCXrt3dTextDrawOptions	XtRBoolean
XtNxrt3dTextStrings XRT3D_TEXT_STRINGS	XtCXrt3dStrings	XtRXrt3dStrings
XtNxrt3dTextStrokeFont XRT3D_TEXT_STROKE_FONT	XtCXrt3dStrokeFont	XtRXrt3dStrokeFont
XtNxrt3dTextStrokeSize XRT3D_TEXT_STROKE_SIZE	XtCXrt3dStrokeFontSize	XtRDimension

G

リソース・ファイルの使用

構文

リソース・ファイルの例

リソース・ファイル、GUIビルダ、*Xrt3dSetPropString()*に使用されるリソースの構文の一覧です。この章の最後には、サンプルのリソース・ファイルが提供されています。XRT/3dは、XtとMotifに定義されていないデータ型のリソース変換のみを定義します。

`UseDefault`を除いて全てのXRT/3dのリソースは、リソース・ファイルで定義できます。また、XRT/3dのテキスト・オブジェクトの全てのリソースもリソース・ファイルで定義できます。

G.1 構文

Comments

行にあるエクスクラメーション・マーク(!)の後のテキストは無視されます。

Continuation(継続)

行の最後の文字をバック・スラッシュ(\)にすると、次に続く行に継続します。

Booleans

大文字、小文字、大小混合文字の“`True`”または“`False`”です。

列挙された型

列挙型を用いるリソースは、列挙された値に基づく一貫した命名規則を使用します。全てのリソースでは、先頭の`XRT_`がなく、残りのアンダー・スコアもありません。リソース値は、大文字、小文字または大小混合の文字で入力できます。例えば、次の各リソースの指定は、どれも`XRT3D_BAR_FIXED`が`XtNxrt3dXBarFormat`で使用されるようにしています。

```
*.XtXrt3d.xrt3dXBarFormat: BarFixed
*.XtXrt3d.xrt3dXBarFormat: BARFIXED
*.XtXrt3d.xrt3dXBarFormat: barfixed
```

データ

XtRXrt3dData型のリソースは、有効なファイル名で指定されます。(環境変数を使用することができます。)リソース変換プログラムは、このファイルを開き、*Xrt3dMakeDataFromFile()*を呼び出します。エラー・メッセージは、*stderr*に書かれます。

```
*.XtXrt3d.xrt3dData: $XRTHOME/src/3d/data/4dbar.dat
```

XtRXrt3dDouble

リソース変換プログラムは、ANSIの関数*strtod()*を呼び出し、リソース指定をdoubleに変換します。

等高線スタイル

XtRXrt3dContourStyles型のリソースは、有効なファイル名で指定されます。リソース変換プログラムはこのファイルを開き、*Xrt3dContourStylesFromFile()*を呼び出します。エラー・メッセージは、*stderr*に書かれます。

分布表(Distribution Table)

XtRXrt3dDistnTable型のリソースは、有効なファイル名で指定されます。リソース変換プログラムはこのファイルを開き、*Xrt3dDistnTableFromFile()*を呼び出します。エラー・エメッセージは、*stderr*に書かれます。

グリッド線

XtRXrt3dGridLines型のリソースは、XYPlane、XZPlane、YZPlaneのストリングの組み合わせ(+を使用)で指定されます。(例)

```
*.XtXrt3d.xrt3dXGridLines: XYPlane + XZPlane
```

イメージ

XtRXrt3dImage型のリソースは、有効なファイル名で指定されます。ファイル名からイメージ型が調べられ、該当するイメージに変換されます。イメージ型はデフォルトのほかに、新たに登録することができます。(例)

```
*.XtXrt3d.xrt3dLegendImage: $XRTHOME/src/icons/shading.xpm
```

ストリングのリスト

XtRXrt3dStrings型のリソースは、括弧で囲まれたストリングのリストとして指定されます。(各ストリングは引用符で囲まれます。)下の例は、フッターを設定しています。

```
*.XtXrt3d.xrt3dFooterStrings: (“Two” “Lines!”)
```

空のリストを指定する場合は、一組の括弧を用いてください。

値ラベル(Value-label)

XtRXrt3dValueLabels型のリソースは、括弧に囲まれたリストとして指定され、各メンバがそれぞれの括弧の中で値－ラベルを指定します。ラベルのストリングは、引用符に囲まれます。

```
*.XtXrt3d.xrt3dXValueLabels: (\
 (2.3 “Cold”)\
 (3.5 “Warm”)\
 (5.2 “Hot”)\
)
```

変換プログラムは、値の増加に従って値ラベルをソートします。二重のエントリは、任意に削除されます。空のリストを指定するには、一組の括弧を用いてください。

XYColors

XtRXrt3dXYColors型のリソースは、括弧に囲まれたリストとして指定され、各メンバがそれぞれの括弧の中でXYColorを指定します。色名は、引用符に囲まれます。

```
*.XtXrt3d.xrt3dXYColors: (\
 (1 1 “red”)\
 (2 -3 “green”)\
 (6 8 “#FF23AC”)\
)
```

エントリは、リストに現れる順に処理、記憶されます。二重のエントリは、先に現れる方が削除されます。空のリストを指定するには、一組の括弧を用いてください。

Projections

XtRXrt3dProject型のリソースは、ProjectContoursとProjectZonesの組み合わせで指定されます。(+を使用) 次の例は、等高線とゾーンのプロジェクション(投影)をユニット・キューブの下部に表示します。

```
*.XtXrt3d.xrt3dProjectZMin: PorjectContours + ProjectZones
```

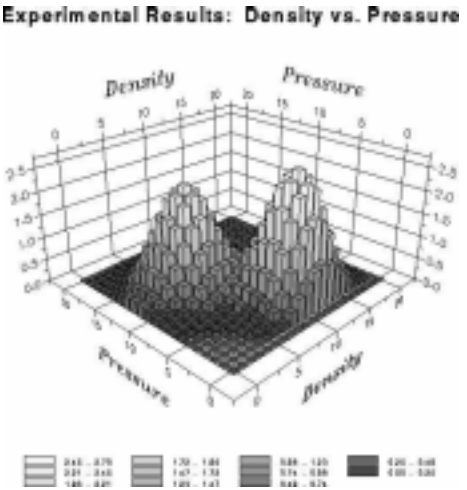
テキスト・オブジェクト

XRT/3dテキスト・オブジェクトのリスト(XtNxrt3dTextList)は、テキスト・オブジェクト名がカンマで区切られたリストとして指定されます。リソース変換プログラムは、指定されたテキスト・オブジェクトを作成します。テキスト・オブジェクトのリソースは、下の例のように、テキスト・オブジェクトの名前を使用して指定できます。

```
*XtXrt3d.xrt3dTextList: text1, text2, text3
*XtXrt3dText.xrt3dTextBackgroundColor: blue
*XtXrt3dText.text2.xrt3dTextBackgroundColor: orange
```

G.2 リソース・ファイルの例

下記のリソース・ファイルは、次の図のグラフを定義しています。




```
! Sample resource file similar to $XRTHOME/src/3d/samples/4dbar.res
*xrt3dDrawMesh: True
*xrt3dRepaint: True
*xrt3dDrawShaded: True
*xrt3dSolidSurface: False
*xrt3dDrawZones: True
*xrt3dSurfaceBottomColor: dim grey
*xrt3dFooterAdjust: AdjustCenter
*xrt3dSurfaceData: $XRTHOME/src/3d/data/4dbar.dat
*xrt3dSurfaceTopColor: light grey
*xrt3dFooterBorder: BorderNone
*xrt3dFooterBorderWidth: 2
*xrt3dType: TypeBar
*xrt3dFooterFont: XtDefaultFont
*xrt3dViewNormalized: True
*xrt3dViewScale: 1
*xrt3dFooterHeight: 0
*height: 582
*xrt3dViewXTranslate: 0
*xrt3dFooterStrings: ()
*xrt3dViewYTranslate: 0
*xrt3dFooterWidth: 0
*xrt3dFooterX: 281
*xrt3dXAnnoMethod: AnnoValues
*xrt3dFooterXUseDefault: True
*xrt3dXAxisShow: True
*xrt3dFooterY: 578
*xrt3dXAxisTitle: Density
*xrt3dFooterYUseDefault: True
*xrt3dXDataLabels: ()
```

```
*xrt3dXGridLines: xyplane+xzplane
*xrt3dForegroundColor: black
*xrt3dGraphBackgroundColor: grey90
*xrt3dXMax: 20
*xrt3dGraphBorder: BorderNone
*xrt3dXMaxUseDefault: False
*xrt3dXMeshFilter: 1
*xrt3dGraphBorderWidth: 2
*xrt3dGraphForegroundColor: black
*xrt3dXMeshShow: True
*xrt3dGraphHeight: 436
*xrt3dXMin: 0
*xrt3dGraphHeightUseDefault: True
*xrt3dXMinUseDefault: False
*xrt3dXRotation: 25.0606
*xrt3dGraphWidth: 512
*xrt3dXScale: 1
*xrt3dGraphWidthUseDefault: True
*xrt3dXValueLabels: ()
*xrt3dGraphX: 25
*xrt3dXYColors: ()
*xrt3dGraphXUseDefault: True
*xrt3dGraphY: 68
*xrt3dYAnnoMethod: AnnoValues
*xrt3dGraphYUseDefault: True
*xrt3dYAxisShow: True
*xrt3dYAxisTitle: Pressure
*xrt3dHeaderAdjust: AdjustCenter
*xrt3dYDataLabels: ()
*xrt3dYGridLines: xyplane+yzplane
*xrt3dHeaderBorder: BorderNone
*xrt3dHeaderBorderWidth: 2
*xrt3dYMax: 20
*xrt3dYMaxUseDefault: True
*xrt3dHeaderFont: -*-helvetica-bold-r-normal--*-240-*-*-*-*-*-*
*xrt3dYMeshFilter: 1
*xrt3dYMeshShow: True
*xrt3dHeaderHeight: 27
*xrt3dHeaderStrings: ("Experimental Results: Density vs. Pressure")
*xrt3dYMin: 0
*xrt3dHeaderWidth: 512
*xrt3dYMinUseDefault: True
*xrt3dYRotation: -0.102286
*xrt3dHeaderX: 25
*xrt3dYScale: 1
*xrt3dHeaderXUseDefault: True
*xrt3dYValueLabels: ()
*xrt3dHeaderY: 22
*xrt3dHeaderYUseDefault: True
*xrt3dZAnnoMethod: AnnoValues
*xrt3dZAxisShow: True
*xrt3dLegendAnchor: AnchorSouth
*xrt3dZGridLines: xzplane+yzplane
*xrt3dLegendBorder: BorderNone
*xrt3dLegendBorderWidth: 2
*xrt3dZMax: 2.69979
*xrt3dZMaxUseDefault: True
*xrt3dLegendFont: -*-helvetica-medium-r-normal--*-100-*-*-*-*-*-*
*xrt3dAxisStrokeFont: SfRomanSimplex
*xrt3dLegendForegroundColor: black
```

```
*xrt3dZMin: 0
*xrt3dAxisStrokeSize: 55
*xrt3dLegendHeight: 48
*xrt3dZMinUseDefault: True
*xrt3dAxisTitleStrokeFont: SfItalicTriplex
*xrt3dZOrigin: 0
*xrt3dAxisTitleStrokeSize: 95
*xrt3dZRotation: 45.3664
*xrt3dLegendOrientation: AlignVertical
*xrt3dZScale: 0.5
*xrt3dLegendShow: True
*xrt3dBackgroundColor: white
*xrt3dZValueLabels: ()
*xrt3dLegendStrings: ()
*xrt3dXBarFormat: BarFixed
*xrt3dLegendStyle: LegendStyleStepped
*xrt3dZoneData: $XRTHOME/src/3d/data/4dbar.dat
*xrt3dXBarSpacing: 80
*xrt3dZoneMethod: ZoneContours
*xrt3dLegendWidth: 480
*xrt3dYBarFormat: BarFixed
*width: 562
*xrt3dLegendX: 39
*xrt3dYBarSpacing: 80
*xrt3dLegendXUseDefault: True
*xrt3dLegendY: 523
*xrt3dDataAreaBackgroundColor: grey80
*xrt3dLegendYUseDefault: True
*xrt3dDebug: False
*xrt3dMeshBottomColor: black
*xrt3dDistnMethod: DistnLinear
*xrt3dMeshTopColor: black
*xrt3dNumDistnLevels: 10
*xrt3dDoubleBuffer: True
*xrt3dPerspectiveDepth: 2.5
*xrt3dDrawContours: True
*xrt3dDrawHiddenLines: False
```

H

UIL引数

XRT/3dのUIL引数とそのデータ型のアルファベット順のリストです。二番めの表は、XRT/3dのUILのreasonです。UILに使用については、8.10を参照してください。

UIL Argument Name	Argument Type
XtNxrt3dAxisStrokeFont	integer
XtNxrt3dAxisStrokeSize	integer
XtNxrt3dAxisTitleStrokeFont	integer
XtNxrt3dAxisTitleStrokeSize	integer
XtNxrt3dBackgroundColor	string
XtNxrt3dDataAreaBackgroundColor	string
XtNxrt3dDataStylesUseDefault	boolean
XtNxrt3dDebug	boolean
XtNxrt3dDistnMethod	integer
XtNxrt3dDoubleBuffer	boolean
XtNxrt3dDrawContours	boolean
XtNxrt3dDrawDropLines	boolean
XtNxrt3dDrawHiddenLines	boolean
XtNxrt3dDrawMesh	boolean
XtNxrt3dDrawShaded	boolean
XtNxrt3dDrawZones	boolean
XtNxrt3dFooterAdjust	integer
XtNxrt3dFooterBackgroundColor	string

UIL Argument Name	Argument Type
XtNxrt3dFooterBorder	integer
XtNxrt3dFooterBorderWidth	integer
XtNxrt3dFooterFont	font_table
XtNxrt3dFooterForegroundColor	string
XtNxrt3dFooterHeight	integer
XtNxrt3dFooterWidth	Integer
XtNxrt3dFooterImage	Pixmap
XtNxrt3dFooterImageLayout	Integer
XtNxrt3dFooterImageMinimumSize	Boolean
XtNxrt3dFooterX	Integer
XtNxrt3dFooterXUseDefault	boolean
XtNxrt3dFooterY	integer
XtNxrt3dFooterYUseDefault	boolean
XtNxrt3dForegroundColor	string
XtNxrt3dGraphBackgroundColor	string
XtNxrt3dGraphBorder	integer
XtNxrt3dGraphBorderWidth	integer
XtNxrt3dGraphForegroundColor	string
XtNxrt3dGraphHeight	integer
XtNxrt3dGraphHeightUseDefault	boolean
XtNxrt3dGraphImage	pixmap
XtNxrt3dGraphImageLayout	integer
XtNxrt3dGraphWidth	integer
XtNxrt3dGraphWidthUseDefault	boolean
XtNxrt3dGraphX	integer
XtNxrt3dGraphXUseDefault	boolean
XtNxrt3dGraphY	integer
XtNxrt3dGraphYUseDefault	boolean
XtNxrt3dHeaderAdjust	integer
XtNxrt3dHeaderBackgroundColor	string
XtNxrt3dHeaderBorder	integer
XtNxrt3dHeaderBorderWidth	integer

UIL Argument Name	Argument Type
XtNxrt3dHeaderFont	font_table
XtNxrt3dHeaderForegroundColor	string
XtNxrt3dHeaderHeight	integer
XtNxrt3dHeaderImage	pixmap
XtNxrt3dHeaderImageLayout	integer
XtNxrt3dHeaderImageMinimumSize	boolean
XtNxrt3dHeaderWidth	integer
XtNxrt3dHeaderX	integer
XtNxrt3dHeaderXUseDefault	boolean
XtNxrt3dHeaderY	integer
XtNxrt3dHeaderYUseDefault	boolean
XtNxrt3dImage	pixmap
XtNxrt3dImageLayout	integer
XtNxrt3dLegendAnchor	integer
XtNxrt3dLegendBackgroundColor	string
XtNxrt3dLegendBorder	integer
XtNxrt3dLegendBorderWidth	integer
XtNxrt3dLegendFont	font_table
XtNxrt3dLegendForegroundColor	string
XtNxrt3dLegendHeight	integer
XtNxrt3dLegendImage	pixmap
XtNxrt3dLegendImageLayout	integer
XtNxrt3dLegendLabelFilter	Integer
XtNxrt3dLegendOrientation	integer
XtNxrt3dLegendShow	boolean
XtNxrt3dLegendStyle	integer
XtNxrt3dLegendTitle	String
XtNxrt3dLegendWidth	integer
XtNxrt3dLegendX	integer
XtNxrt3dLegendXUseDefault	boolean
XtNxrt3dLegendY	integer
XtNxrt3dLegendYUseDefault	boolean

UIL Argument Name	Argument Type
XtNxrt3dMeshBottomColor	string
XtNxrt3dMeshTopColor	string
XtNxrt3dNumDistnLevels	integer
XtNxrt3dPerspectiveDepth	float
XtNxrt3dPreviewMethod	integer
XtNxrt3dProjectZMax	integer
XtNxrt3dProjectZMin	integer
XtNxrt3dRepaint	boolean
XtNxrt3dSolidSurface	boolean
XtNxrt3dSurfaceBottomColor	string
XtNxrt3dSurfaceTopColor	string
XtNxrt3dTextAdjust	integer
XtNxrt3dTextAttachIndexPoint	integer
XtNxrt3dTextAttachIndexSeries	integer
XtNxrt3dTextAttachIndexX	integer
XtNxrt3dTextAttachIndexY	integer
XtNxrt3dTextAttachMethod	integer
XtNxrt3dTextAttachPixelX	integer
XtNxrt3dTextAttachPixelY	integer
XtNxrt3dTextAttachPointX	float
XtNxrt3dTextAttachPointY	float
XtNxrt3dTextAttachPointZ	float
XtNxrt3dTextBackgroundColor	string
XtNxrt3dTextBorder	integer
XtNxrt3dTextBorderWidth	integer
XtNxrt3dTextFont	font_table
XtNxrt3dTextForegroundColor	string
XtNxrt3dTextImage	pixmap
XtNxrt3dTextImageLayout	integer
XtNxrt3dTextImageMinimumSize	boolean
XtNxrt3dTextLineShow	boolean
XtNxrt3dTextOffsetX	integer

UIL Argument Name	Argument Type
XtNxrt3dTextOffsetY	integer
XtNxrt3dTextPlane	integer
XtNxrt3dTextPrintFont	string
XtNxrt3dTextShow	boolean
XtNxrt3dTextStrokeFont	integer
XtNxrt3dTextStrokeSize	integer
XtNxrt3dType	integer
XtNxrt3dViewNormalized	boolean
XtNxrt3dViewScale	float
XtNxrt3dViewXTranslate	float
XtNxrt3dViewYTranslate	float
XtNxrt3dXAnnoMethod	integer
XtNxrt3dXAxisShow	boolean
XtNxrt3dXAxisTitle	string
XtNxrt3dXBarFormat	integer
XtNxrt3dXBarSpacing	float
XtNxrt3dXGridLines	integer
XtNxrt3dXGridLineStyleUseDefault	boolean
XtNxrt3dXMax	float
XtNxrt3dXMaxUseDefault	boolean
XtNxrt3dXMeshFilter	integer
XtNxrt3dXMeshShow	boolean
XtNxrt3dXMin	float
XtNxrt3dXMinUseDefault	boolean
XtNxrt3dXRotation	float
XtNxrt3dXScale	float
XtNxrt3dYAnnoMethod	integer
XtNxrt3dYAxisShow	boolean
XtNxrt3dYAxisTitle	string
XtNxrt3dYBarFormat	integer
XtNxrt3dYBarSpacing	float
XtNxrt3dYGridLines	integer

UIL Argument Name	Argument Type
XtNxrt3dYGridLineStyleUseDefault	boolean
XtNxrt3dYMax	float
XtNxrt3dYMaxUseDefault	boolean
XtNxrt3dYMeshFilter	integer
XtNxrt3dYMeshShow	boolean
XtNxrt3dYMin	float
XtNxrt3dYMinUseDefault	boolean
XtNxrt3dYRotation	float
XtNxrt3dYScale	float
XtNxrt3dZAnnoMethod	integer
XtNxrt3dZAxisShow	boolean
XtNxrt3dZAxisTitle	string
XtNxrt3dZGridLines	integer
XtNxrt3dZGridLineStyleUseDefault	boolean
XtNxrt3dZMax	float
XtNxrt3dZMaxUseDefault	boolean
XtNxrt3dZMin	float
XtNxrt3dZMinUseDefault	boolean
XtNxrt3dZoneMethod	integer
XtNxrt3dZOrigin	float
XtNxrt3dZRotation	float
XtNxrt3dZScale	float

Reasons
XtNxrt3dExposeCallback
XtNxrt3dMapCallback
XtNxrt3dModifyCallback
XtNxrt3dPickCallback
XtNxrt3dResizeCallback
XtNxrt3dRotateCallback
XtNxrt3dTransformCallback

索引

4

4Dサーフェス・グラフ, 87
4D棒グラフ, 88

C

C++, 11, 101

H

hole, 12, 64

M

MS Windowsへのポーティング, 98

N

noval, 64, 66

U

UIL, 101, 223
`UseDefault`, 18

X

XRT/3d
　用語の解説, 15
XRT/3d 3.0
　変更点, 2
xrt_auth, 10
Xrt3dAdjust, 193
Xrt3dAlign, 193
Xrt3dAnchor, 193
Xrt3dAnnoMethod, 193
Xrt3dArgValToDouble(), 161
Xrt3dAxis, 194
Xrt3dBarFormat, 194
Xrt3dBorder, 194
Xrt3dCallbackStruct, 194
Xrt3dColor, 194
Xrt3dComputeZValue(), 83, 161
Xrt3dContourStyle, 95, 195
Xrt3dContourStylesFromFile(), 162
Xrt3dData, 12, 63, 66, 87, 195
Xrt3dDataCopy(), 162
Xrt3dDataShaded(), 162
Xrt3dDataSmooth(), 163
Xrt3dDataStyle, 58, 195
Xrt3dDataType, 195
Xrt3dDataUpdateDataValue(), 163
Xrt3dDataWindow(), 163
Xrt3dDestroyData(), 164
Xrt3dDistnIndex(), 164
Xrt3dDistnMethod, 195
Xrt3dDistnRange, 195
Xrt3dDistnTable, 196
Xrt3dDistnTableFromFile(), 164
Xrt3dDoubleToArgVal(), 17, 165
Xrt3dDrawCGM(), 41, 165
Xrt3dDrawPS(), 40, 166
Xrt3dDrawPSColor, 196
Xrt3dDupContourStyles(), 168
Xrt3dDupDataStyles(), 168
Xrt3dDupDistnTable(), 168
Xrt3dDupStrings(), 168
Xrt3dDupValueLabels(), 169
Xrt3dDupXYColors(), 169
Xrt3dEnableImageType(), 169
Xrt3dFreeContourStyles(), 169
Xrt3dFreeDataStyles(), 169
Xrt3dFreeDistnTable(), 170
Xrt3dFreePropString(), 170
Xrt3dFreeStrings(), 170
Xrt3dFreeValueLabels(), 170
Xrt3dFreeXYColors(), 170
Xrt3dGetDistnTable(), 170
Xrt3dGetImage(), 41, 171
Xrt3dGetNthContourStyle(), 96, 171
Xrt3dGetNthDataLabel(), 171
Xrt3dGetNthDataStyle(), 171
Xrt3dGetNthFooterString(), 172
Xrt3dGetNthHeaderString(), 172
Xrt3dGetNthLegendString(), 172
Xrt3dGetPropString(), 172
Xrt3dGetValueLabel(), 173
Xrt3dGetValues(), 173

Xrt3dGetXYColor(), 50, 173
Xrt3dGridData, 66, 196
Xrt3dImage, 196
Xrt3dImageLayout, 196
Xrt3dInsertNthDataStyle(), 174
Xrt3dInterpMethod, 196
Xrt3dIrGridData, 67, 197
Xrt3dLegendStyle, 197
Xrt3dLinePattern, 197
Xrt3dLineStyle, 197
Xrt3dMakeDataFromFile(), 64, 174
Xrt3dMakeGridData(), 12, 64, 176
Xrt3dMakeIrGridData(), 176
Xrt3dMakePointData(), 176
Xrt3dMakePointDataSeries(), 177
Xrt3dMap(), 78, 79, 82, 177
Xrt3dMapCallbackStruct, 197
Xrt3dMapResult, 198
Xrt3dModifyCallbackStruct, 198
Xrt3dMrmInitialize(), 177
Xrt3dOutputPS(), 41, 177
Xrt3dOutputXwd(), 41, 178
Xrt3dPick(), 78, 80, 178
Xrt3dPickCallbackStruct, 198
Xrt3dPickResult, 80, 198
Xrt3dPoint3D, 198
Xrt3dPointData, 198
Xrt3dPointSeries, 199
Xrt3dPreviewMethod, 199
Xrt3dProperty, 199
Xrt3dReason, 199
Xrt3dRegion, 199
Xrt3dRemoveNthDataStyle(), 178
Xrt3dResetContourStyles(), 97, 179
Xrt3dRotateCallbackStruct, 199
Xrt3dSampleColorFunc(), 179
Xrt3dSaveDataToFile(), 179
Xrt3dSaveImageAsJpeg(), 179
Xrt3dSaveImageAsPng(), 180
Xrt3dSaveImageGeneric(), 180
Xrt3dSetNthContourStyle(), 96, 180
Xrt3dSetNthDataLabel(), 181
Xrt3dSetNthDataStyle(), 181
Xrt3dSetNthFooterString(), 181
Xrt3dSetNthHeaderString(), 182
Xrt3dSetNthLegendString(), 182
Xrt3dSetPropString(), 182
Xrt3dSetReverseConverters(), 102, 182
Xrt3dSetValueLabel(), 183
Xrt3dSetValues(), 183
Xrt3dSetXYColor(), 50, 183
Xrt3dStrokeFont, 200
Xrt3dSymbolPattern, 202
Xrt3dSymbolStyle, 202
Xrt3dTextAttachMethod, 203
Xrt3dTextGetPropString(), 184
Xrt3dTextGetValues(), 184
Xrt3dTextSetPropString(), 184
Xrt3dTextSetValues(), 185
Xrt3dTransformCallbackStruct, 203
Xrt3dType, 203
Xrt3dUnmap(), 83, 185
Xrt3dUnpick(), 82, 185
Xrt3dValueLabel, 203
Xrt3dXYColor, 50, 203
Xrt3dZInterpolate(), 186
Xrt3dZoneMethod, 203
XRTHOME, 9
XrtMapResult, 83
XrtPDSCmpDoubles(), 116, 120
XrtPDSCmpFloats(), 116, 120
XrtPDSCmpInts(), 116, 121
XrtPDSCmpStrings(), 116, 121
XrtPDSCmpStringsIgnoreCase(), 121
XrtPDSImage, 105
XrtPDSImageCopy(), 118
XrtPDSImageCreate(), 105, 118
XrtPDSImageCreateProc, 109
XrtPDSImageDestroy(), 118
XrtPDSImageFuncs, 106
XrtPDSImageGetClipMask(), 119
XrtPDSImageGetColorDepth(), 119
XrtPDSImageGetFilename(), 118
XrtPDSImageGetHeight(), 119
XrtPDSImageGetImage(), 119
XrtPDSImageGetWidth(), 119
XrtPDSImageHeaderCompareProc, 109
XrtPDSImageIsClipMaskSet(), 120
XrtPDSImageIsImage(), 120
XrtPDSImageIsSet(), 119
XrtPDSImageNameCompareProc, 109
XrtPDSImageOutputProc, 109
XrtPDSList, 110
XrtPDSListAdd(), 112
XRTPDSListAdd(), 121
XrtPDSListAppend(), 111, 122
XrtPDSListCreate(), 111, 122
XrtPDSListDelete(), 112, 122
XrtPDSListDeleteALL(), 122
XrtPDSListDestroy(), 123
XrtPDSListDestroyObject(), 123
XrtPDSListDestroyXtPoiter(), 123
XrtPDSListDuplicate(), 123
XrtPDSListFind(), 113, 123

XrtPDSListGetCompareProc(), 124
XrtPDSListGetItem(), 114, 124
XrtPDSListGetItemCount(), 115, 124
XrtPDSListGetItems(), 124
XrtPDSListGetItemSize(), 124
XrtPDSListGetUserData(), 125
XrtPDSListInsertAfter(), 125
XrtPDSListInsertBefore(), 125
XrtPDSListInsetBefore(), 111
XrtPDSListIsList(), 113, 125
XrtPDSListIsSorted(), 126
XrtPDSListLookupItem(), 113, 126
XrtPDSListMoveItemsAfter(), 126
XrtPDSListMoveItemsBefore(), 117, 126
XrtPDSListRemove(), 112, 127
XrtPDSListRemoveAll(), 127
XrtPDSListSearchForward(), 115, 127
XrtPDSListSearchReverse(), 128
XrtPDSListSetCompareProc(), 113, 128
XrtPDSListSetDestroyProc(), 112, 128
XrtPDSListSetItem(), 129
XrtPDSListSetUserData(), 117, 129
XrtPDSListSort(), 116, 129
XrtSaveImageGeneric(), 108
XtAddTimeOut(), 65
XtAppAddInput(), 65
XtCreateXrt3d(), 186
XtCreateXrt3dText(), 186
XtIsXrt3d(), 186
XtIsXrt3dText(), 186
XtNxrt3d[XY]BarSpacing, 48, 152
XtNxrt3d[XY]Max, 29
XtNxrt3d[XY]MeshFilter, 54
XtNxrt3d[XY]Min, 29
XtNxrt3d[XYZ]AnnoMethod, 32
XtNxrt3d[XYZ]AxisShow, 29
XtNxrt3d[XYZ]GridLines, 33
XtNxrt3d[XYZ]GridLineStyle, 153
XtNxrt3d[XYZ]GridLineStyleUseDefault, 153
XtNxrt3d[XYZ]Rotation, 76
XtNxrt3d[XYZ]Scale, 31
XtNxrt3d[XYZ]Title, 29
XtNxrt3dAxisStrokeFont, 29, 134
XtNxrt3dAxisStrokeSize, 134
XtNxrt3dAxisTitleStrokeFont, 134
XtNxrt3dAxisTitleStrokeSize, 134
XtNxrt3dBackgroundColor, 36, 134
XtNxrt3dColorFunc, 37
XtNxrt3dContourStyles, 95, 96, 134
XtNxrt3dDataAreaBackgroundColor, 135
XtNxrt3dDataStyles, 135
XtNxrt3dDataStylesUseDefault, 135
XtNxrt3dDebug, 21, 135
XtNxrt3dDistnMethod, 93, 135
XtNxrt3dDistnTable, 93, 136
XtNxrt3dDoubleBuffer, 136
XtNxrt3dDrawContours, 136
XtNxrt3dDrawDropLines, 136
XtNxrt3dDrawHiddenLines, 55, 65, 136
XtNxrt3dDrawMesh, 137
XtNxrt3dDrawShaded, 137
XtNxrt3dDrawZones, 137
XtNxrt3dExposeCallback, 137
XtNxrt3dFooterAdjust, 33, 138
XtNxrt3dFooterBackgroundColor, 138
XtNxrt3dFooterBorder, 138
XtNxrt3dFooterBorderWidth, 139
XtNxrt3dFooterFont, 139
XtNxrt3dFooterForegroundColor, 139
XtNxrt3dFooterHeight, 139
XtNxrt3dFooterImage, 139
XtNxrt3dFooterImageLayout, 139
XtNxrt3dFooterImageMinimumSize, 139
XtNxrt3dFooterStrings, 33, 139
XtNxrt3dFooterWidth, 139
XtNxrt3dFooterX, 139
XtNxrt3dFooterXUseDefault, 139
XtNxrt3dFooterY, 139
XtNxrt3dFooterYUseDefault, 139
XtNxrt3dForegroundColor, 37, 140
XtNxrt3dGraphBackgroundColor, 140
XtNxrt3dGraphBorder, 36, 140
XtNxrt3dGraphBorderWidth, 36, 140
XtNxrt3dGraphForegroundColor, 140
XtNxrt3dGraphHeight, 140
XtNxrt3dGraphHeightUseDefault, 141
XtNxrt3dGraphImage, 140
XtNxrt3dGraphImageLayout, 140
XtNxrt3dGraphWidth, 140
XtNxrt3dGraphWidthUseDefault, 141
XtNxrt3dGraphX, 141
XtNxrt3dGraphXUseDefault, 141
XtNxrt3dGraphY, 141
XtNxrt3dGraphYUseDefault, 141
XtNxrt3dHeaderAdjust, 33, 141
XtNxrt3dHeaderBackgroundColor, 141
XtNxrt3dHeaderBorder, 141
XtNxrt3dHeaderBorderWidth, 141
XtNxrt3dHeaderFont, 141
XtNxrt3dHeaderForegroundColor, 37, 142
XtNxrt3dHeaderHeight, 142
XtNxrt3dHeaderImage, 142
XtNxrt3dHeaderImageLayout, 142
XtNxrt3dHeaderImageMinimumSize, 142

XtNxrt3dHeaderStrings, 33, 142
XtNxrt3dHeaderWidth, 142
XtNxrt3dHeaderX, 142
XtNxrt3dHeaderXUseDefault, 142
XtNxrt3dHeaderY, 142
XtNxrt3dHeaderYUseDefault, 142
XtNxrt3dImage, 142
XtNxrt3dImageLayout, 143
XtNxrt3dLegendAnchor, 30, 143
XtNxrt3dLegendBackgroundColor, 143
XtNxrt3dLegendBorder, 143
XtNxrt3dLegendBorderWidth, 143
XtNxrt3dLegendDistnRange, 143
XtNxrt3dLegendFont, 143
XtNxrt3dLegendForegroundColor, 143
XtNxrt3dLegendHeight, 145
XtNxrt3dLegendImage, 143
XtNxrt3dLegendImageLayout, 144
XtNxrt3dLegendLabelFilter, 144
XtNxrt3dLegendLabelFunc, 94, 144
XtNxrt3dLegendOrientation, 30, 144
XtNxrt3dLegendShow, 29, 144
XtNxrt3dLegendStrings, 94, 144
XtNxrt3dLegendStyle, 30, 145
XtNxrt3dLegendTitle, 145
XtNxrt3dLegendWidth, 145
XtNxrt3dLegendX, 145
XtNxrt3dLegendXUseDefault, 145
XtNxrt3dLegendY, 145
XtNxrt3dLegendYUseDefault, 145
XtNxrt3dMapCallback, 145
XtNxrt3dMeshBottomColor, 54, 146
XtNxrt3dMeshTopColor, 54, 146
XtNxrt3dModifyCallback, 74, 78, 146
XtNxrt3dNumDistnLevels, 93, 147
XtNxrt3dPerspectiveDepth, 31, 147
XtNxrt3dPickCallback, 147
XtNxrt3dPreviewMethod, 147
XtNxrt3dProjectZMax, 52, 148
XtNxrt3dProjectZMin, 52, 148
XtNxrt3dRepaint, 65, 148
XtNxrt3dResizeCallback, 148
XtNxrt3dRotateCallback, 76, 148
XtNxrt3dSolidSurface, 55, 149
XtNxrt3dSurfaceBottomColor, 55, 149
XtNxrt3dSurfaceData, 18, 65, 149
XtNxrt3dSurfaceTopColor, 55, 149
XtNxrt3dTextAdjust, 91, 157
XtNxrt3dTextAttachIndex[XY], 91
XtNxrt3dTextAttachIndexPoint, 158
XtNxrt3dTextAttachIndexSeries, 158
XtNxrt3dTextAttachIndexX, 158
XtNxrt3dTextAttachIndexY, 158
XtNxrt3dTextAttachMethod, 89, 91, 158
XtNxrt3dTextAttachPixel[XY], 92
XtNxrt3dTextAttachPixelX, 158
XtNxrt3dTextAttachPixelY, 158
XtNxrt3dTextAttachPoint[XYZ], 91
XtNxrt3dTextAttachPointX, 158
XtNxrt3dTextAttachPointY, 158
XtNxrt3dTextAttachPointZ, 158
XtNxrt3dTextBackgroundColor, 158
XtNxrt3dTextBorder, 92, 158
XtNxrt3dTextBorderWidth, 92, 159
XtNxrt3dTextFont, 159
XtNxrt3dTextForegroundColor, 159
XtNxrt3dTextImage, 159
XtNxrt3dTextImageLayout, 159
XtNxrt3dTextImageMinimumSize, 159
XtNxrt3dTextLineShow, 92, 159
XtNxrt3dTextList, 149
XtNxrt3dTextOffset[XY], 91
XtNxrt3dTextOffsetX, 159
XtNxrt3dTextOffsetY, 159
XtNxrt3dTextPlane, 91, 160
XtNxrt3dTextShow, 93, 160
XtNxrt3dTextStrings, 91, 160
XtNxrt3dTextStrokeFont, 160
XtNxrt3dTextStrokeSize, 92, 160
XtNxrt3dTransformCallback, 75, 149
XtNxrt3dTtextPrintFont, 160
XtNxrt3dType, 43, 150
XtNxrt3dView[XY]Translate, 74
XtNxrt3dViewNormalized, 76, 151
XtNxrt3dViewScale, 74, 151
XtNxrt3dViewXTranslate, 151
XtNxrt3dViewYTranslate, 151
XtNxrt3dXAnnoMethod, 152
XtNxrt3dXAxisShow, 152
XtNxrt3dXAxisTitle, 152
XtNxrt3dXBarFormat, 152
XtNxrt3dXDataLabels, 152
XtNxrt3dXGridLines, 153
XtNxrt3dXMax, 153
XtNxrt3dXMaxUseDefault, 153
XtNxrt3dXMeshFilter, 154
XtNxrt3dXMeshShow, 154
XtNxrt3dXMin, 154
XtNxrt3dXMinUseDefault, 154
XtNxrt3dXRotation, 154
XtNxrt3dXScale, 155
XtNxrt3dXValueLabels, 155
XtNxrt3dXYColors, 50, 155
XtNxrt3dYAnnoMethod, 152

XtNxrt3dYAxisShow, 152
XtNxrt3dYAxisTitle, 152
XtNxrt3dYBarFormat, 152
XtNxrt3dYDataLabels, 152
XtNxrt3dYGridLines, 153
XtNxrt3dYMax, 153
XtNxrt3dYMaxUseDefault, 153
XtNxrt3dYMeshFilter, 154
XtNxrt3dYMeshShow, 154
XtNxrt3dYMin, 154
XtNxrt3dYMinUseDefault, 154
XtNxrt3dYRotation, 154
XtNxrt3dYScale, 155
XtNxrt3dYValueLabels, 155
XtNxrt3dZAnnoMethod, 152
XtNxrt3dZAxisShow, 152
XtNxrt3dZAxisTitle, 152
XtNxrt3dZGridLines, 153
XtNxrt3dZMax, 153
XtNxrt3dZMin, 154
XtNxrt3dZoneData, 18, 87, 155
XtNxrt3dZoneMethod, 53, 155
XtNxrt3dZOrigin, 48, 155
XtNxrt3dZRotation, 154
XtNxrt3dZScale, 155
XtNxrt3dZValueLabels, 155
XtNxrtDoubleBuffer, 40
XtNxrtExposeCallback, 84
XtNxrtResizeCallback, 84
xtXrt3dTextObject, 89

あ

アクション・ルーチン
  DrawGraph(), 74, 188
  DrawPreview(), 74, 188
  Map(), 79, 188
  ModifyCancel(), 188
  ModifyEnd(), 78, 189
  ModifyStart(), 73, 188
  NoAction(), 77, 189
  Pick(), 78, 189
  RotateUpdate(), 76, 189
  ScaleUpdate(), 74, 190
  SelectAxis(), 190
  TranslateUpdate(), 74, 190
  ViewReset(), 76, 190
  ZoomCancel(), 191
  ZoomEnd(), 74, 191
  ZoomUpdate(), 190
アプリケーションの許可, 10
イメージ
  型の登録, 106
  背景, 38, 92
  フォーマット, 41
色, 19
  Xrt3dXYColor, 50
  カラーマップのフラッシュ, 20
  サーフェス, 55
  指定, 20
  前景色, 37
  背景色, 36
  棒グラフ, 50
  メッシュ, 54
印刷, 40
隠線の消去, 55
遠近法, 31

か

環境変数, 9
グラフの種類, 43
  4Dグラフ, 87
  DrawContours, 43
  DrawMesh, 43
  DrawShaded, 43
  DrawZones, 43
  XtNxrt3dType, 43
  スキャッター・プロット, 57
  ヒストグラム, 49
  棒グラフ, 48
グリッド
  間隔(step), 12
  規則的, 66
  表示, 33
  不規則的, 67
原点, 48
国際化サポート, 98
コールバック
  Expose, 84, 137
  Map, 145, 197
  Modify, 73, 146, 198
  Pick, 78, 147, 198
  reason, 199
  Resize, 84, 148
  Rotate, 76, 148
  Transform, 74, 75, 149, 203

さ

サーフェス
  色, 55
  側面, 55
サンプル・プログラム, 13
軸
  最大値と最小値, 29
  スケーリング, 31
  タイトル, 29
  注釈, 32
  表示, 29
  フォント, 29
ゾーン
  表示法, 53
  プロジェクション, 52

た

ダブル・バッファリング, 40
テキスト・オブジェクト, 89
  位置決め, 91
  色, 92
  型, 90
  削除, 93
  作成, 90
  接続線, 92
  テキスト, 91
  貼り付け, 90
  フォント, 92
  リソース, 90, 157, 213
  枠線, 92
データ, 193
  hole, 12, 63, 88
  `Xrt3dData`, 63
  ガイドライン, 63
  簡易手続き, 69
  規則的なグリッド, 66
  スタイル, 58
  不規則的なグリッド, 67
  変更, 65
  リアルタイム, 65
  ロード, 12, 63, 64
デバッグ, 21
等高線, 52
  カスタマイズ, 95
  スタイル, 52, 96
  プロジェクション, 52
トランスレーション・テーブル, 187
ドロップ・ライン, 58

は

凡例
  位置(アンカー), 30
  カスタマイズ, 94
  スタイル, 30
  ストリング, 94
  表示, 29
  方向, 30
ヒストグラム, 49
フォント, 167
  軸, 29
  ヘッダー、フッター、凡例, 34
フッター, 33
分布表, 93
ヘッダー, 33
棒グラフ
  色, 50
  原点, 48
  ユーザ・インタラクション, 84

ま

メッシュ
  色, 54
  フィルタ, 54

や

ユーザ・インタラクション, 71
  アップデート, 74
  移動, 74
  イベント処理, 80
  開始, 73
  回転, 76
  カスタマイズ, 73
  グラフ・データ, 78
  コントロール, 75
  終了, 78
  スケーリング, 74
  ズーム, 74

デフォルト, 71
トランスレーション・テーブル, 187
不可, 74, 77
棒グラフ, 84
リサイズ, 84
リセット, 76
ユニット・キューブ, 15

ら

リアルタイム
パフォーマンス, 65
リソース
`UseDefault`, 18
位置決め, 35
一覧, 23, 133
型とクラス, 205
バッチ処理, 19
浮動小数点, 17
変換, 102
ポインタ, 18
リソース・ファイル, 21
使用法, 217
例, 220
領域の位置, 35

わ

枠線, 36