Motif 対応 XRT/gauge™

プログラマーズ・ガイド&リファレンス・マニュアル

UNIX Edition ■ Version 1.1

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

2000 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©2000 (株)タンジェント・コンピューティング

XRT、XRT/3d、XRT/care、XRT/field、XRT/gauge、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"に次ページ参照にご従います。

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/gauge について ····· 1
XRT/gauge ウィジェット・ファミリ ····· 1
XRT/gauge オブジェクト ····· 1
前提知識 ····· 2
表記上の規約 ····· 2
関連資料 ····· 2
テクニカル・サポート ····· 2
お客様のご意見 ····· 3

Part I　XRT/gauge の使用法

1　XRT/gauge について ····· 7

1.1　XRT/gauge とは ····· 7
1.2　XRT/gauge と Motif のクラス階層 ····· 9
1.3　環境変数 ····· 9
1.4　プログラムのコンパイルと許可 ····· 11
1.5　リソースファイルと GUI ビルダ ····· 12
1.6　XRT/gauge で C++を使用する ····· 12
1.7　XRT/gauge の国際化 ····· 13
1.8　XRT/gauge の共通なエレメント ····· 14
1.8.1　マクロ ····· 14
1.8.2　Fallback リソースの設定 ····· 14
1.8.3　リソース・アップデートのバッチ処理 ····· 14
1.8.4　デバッグ ····· 15
1.8.5　テキスト・オブジェクトのセンタリング ····· 15

2　Circular Gauge と Circular Dial ウィジェット ····· 17

2.1　ウィジェットの解説 ····· 17
2.1.1　Circular Gauge と Circular Dial に共通なエレメント ····· 18
2.1.2　Circular Gauge ····· 20
2.1.3　Circular Dial ····· 20

2.2 Circular Gauge と Circular Dial のオブジェクト ····· 21
2.2.1 Center オブジェクト(Circular Gauge) ····· 21
2.2.2 Circular Scale(Circular Gauge または Circular Dial) ····· 21
2.2.3 Frame ····· 22
2.2.4 Marker(Dial) ····· 22
2.2.5 Needle(Circular Gauge) ····· 22
2.2.6 Range ····· 23
2.2.7 Switch ····· 23
2.2.8 State ····· 23
2.2.9 テキスト領域 ····· 24
2.2.10 Tick(Gauge) ····· 24
2.3 トランスレーションとアクション ····· 24
2.4 サンプル・コード ····· 25
2.5 ウィジェットの概要 ····· 27
2.6 リソースの一覧 ····· 28
2.6.1 両方の Circular ウィジェットに共通なリソース ····· 28
2.6.2 Circular Gauge – その他のリソース ····· 30
2.6.3 Circular Dial – その他のリソース ····· 30
2.7 手続きとマクロ ····· 32

3 Linear Gauge と Linear Dial ウィジェット ····· 35

3.1 ウィジェットについて ····· 35
3.1.1 Linear Gauge と Linear Dial に共通なエレメント ····· 36
3.1.2 Linear Gauge オブジェクト ····· 37
3.1.3 Linear Dial オブジェクト ····· 38
3.2 サンプル・プログラム ····· 39
3.3 トランスレーションとアクション ····· 40
3.4 ウィジェットの概要 ····· 40
3.5 リソースの一覧 ····· 41
3.5.1 共通なリソース ····· 41
3.5.2 Linear Gauge – その他のリソース ····· 43
3.5.3 Linear Dial – その他のリソース ····· 43
3.6 手続きとマクロ ····· 44

4 LEDIcon ウィジェット ····· 47

4.1 はじめに ····· 47
4.2 LEDIcon のリソース ····· 48
4.3 トランスレーションとアクション ····· 50
4.4 サンプル・コード ····· 50
4.5 ウィジェットの概要 ····· 50
4.6 リソースの一覧 ····· 51
4.7 手続きとマクロ ····· 55

5 LEDNumerical ウィジェット ····· 57

5.1 はじめに ····· 57
5.2 LEDNumerical ウィジェットの値 ····· 58
5.2.1 小数点 ····· 58
5.3 ウィジェットと表示セグメントの色 ····· 58
5.4 サイズ ····· 58
5.5 クロック表示 ····· 60
5.6 数値の傾き ····· 60
5.7 コールバック ····· 61
5.8 サンプル･コード ····· 61
5.9 ウィジェットの概要 ····· 61
5.10 リソースの一覧 ····· 62
5.11 手続きとマクロ ····· 65

6 Percent ウィジェット ····· 67

6.1 Percent ウィジェット ····· 67
6.2 Scale の選択 ····· 69
6.3 ラベル ····· 70
6.4 Item 上にラベルを配置 ····· 71
6.5 Percent ウィジェットのリソースについての注意 ····· 71
6.6 トランスレーションとコールバック ····· 72
6.7 ウィジェットの概要 ····· 72
6.8 リソースの一覧 ····· 72
6.9 手続きとマクロ ····· 75

7 Center オブジェクト ····· 77

7.1 XrtGaCenter ····· 77
7.2 Center オブジェクトのリソースに関する注意 ····· 77
7.3 サンプル･コード ····· 78
7.4 リソースの一覧 ····· 79
7.5 手続きとマクロ ····· 79

8 Circular Scale オブジェクト ····· 81

8.1 XrtGaCircularScale ····· 81
8.2 Circular Scale の角度 ····· 81
8.3 Circular Scale のリソースについて ····· 82
8.3.1 方向の指定 ····· 83
8.3.2 開始と終了の角度の指定 ····· 83
8.3.3 コールバック ····· 83

8.4 サンプル･コード ····· 83
8.5 リソースの一覧 ····· 84
8.6 手続きとマクロ ····· 85

9 Counter オブジェクト ····· 87

9.1 XrtGaCounter ····· 87
9.2 Counter リソース ····· 87
9.2.1 Counter オブジェクトをウィジェットとして使用する ･ 87
9.2.2 枠線と背景の色の選択 ····· 87
9.2.3 色の選択 ····· 88
9.2.4 フォントの選択 ····· 88
9.2.5 背景にイメージを使用 ····· 88
9.2.6 小数点の設定 ····· 88
9.2.7 Transitions の使用 ····· 88
9.2.8 リセット・ボタン ····· 88
9.2.9 Counter への値の設定 ····· 89
9.3 コールバックとトランスレーション ····· 89
9.4 サンプル･コード ····· 89
9.5 ウィジェットの概要 ····· 90
9.6 リソースの一覧 ····· 90
9.7 手続きとマクロ ····· 93

10 Frame オブジェクト ····· 95

10.1 XrtGaFrame ····· 95
10.2 Frame オブジェクトのリソースについて ····· 96
10.3 サンプル･コード ····· 96
10.4 リソースの一覧 ····· 96
10.5 手続きとマクロ ····· 98

11 Icon オブジェクト ····· 99

11.1 XrtGaIcon ····· 99
11.2 Icon のリソースについて ····· 99
11.2.1 Icon の色 ····· 99
11.2.2 Icon にイメージを使用する ····· 100
11.2.3 Icon の形 ····· 100
11.3 サンプル･コード ····· 100
11.4 リソースの一覧 ····· 100
11.5 手続きとマクロ ····· 101

12 Indicator オブジェクト ····· 103

12.1 XrtGaIndicator ····· 103

12.2 Indicator のリソースについて ····· 104
12.2.1 Indicator のリソース ····· 104
12.2.2 Indicator のサイズ ····· 104
12.2.3 ポインタやトライアングルの方向を逆にする ····· 104
12.3 コールバック ····· 104
12.4 サンプル・コード ····· 104
12.5 リソースの一覧 ····· 105
12.6 手続きとマクロ ····· 107

13 Item オブジェクト ····· 109

13.1 XrtGaItem ····· 109
13.2 Item オブジェクトのリソースについて ····· 110
13.3 サンプル・コード ····· 111
13.4 リソースの一覧 ····· 111
13.5 手続きとマクロ ····· 112

14 Linear Scale オブジェクト ····· 115

14.1 XrtGaLinearScale ····· 115
14.2 Linear Scale の作成 ····· 115
14.2.1 Linear Scale のサイズと位置 ····· 116
14.3 ユーザ・インタラクション ····· 117
14.4 サンプル・コード ····· 117
14.5 リソースの一覧 ····· 118
14.6 手続きとマクロ ····· 119

15 Marker オブジェクト ····· 121

15.1 XrtGaMarker ····· 121
15.2 Marker のリソースについて ····· 122
15.3 サンプル・コード ····· 122
15.4 リソースの一覧 ····· 123
15.5 手続きとマクロ ····· 124

16 Needle オブジェクト ····· 125

16.1 XmNxrtGaNeedle ····· 125
16.2 Needle オブジェクトのリソースについて ····· 126
16.3 サンプル・コード ····· 126
16.4 リソースの一覧 ····· 127
16.5 手続きとマクロ ····· 128

17 Range オブジェクト ····· 129

17.1 XrtGaRange ····· 129
17.2 Range オブジェクトのリソースについて ····· 130
17.3 サンプル・コード ····· 132
17.4 リソースの一覧 ····· 133
17.5 手続きとマクロ ····· 135

18 State オブジェクト ····· 137

18.1 XrtGaState ····· 137
18.2 State オブジェクトのリソースについて ····· 137
18.3 サンプル・コード ····· 139
18.4 リソースの一覧 ····· 140
18.5 手続きとマクロ ····· 143

19 Switch オブジェクト ····· 145

19.1 XrtGaSwitch ····· 145
19.2 Switch について ····· 146
19.3 サンプル・コード ····· 147
19.4 コールバック ····· 148
19.5 リソースの一覧 ····· 148
19.6 手続きと方法 ····· 150

20 Text オブジェクト ····· 153

20.1 XrtGaText ····· 153
20.2 Text オブジェクトのリソースについて ····· 153
20.3 サンプル・コード ····· 154
20.4 リソースの一覧 ····· 155
20.5 手続きと方法 ····· 157

21 Tick オブジェクト ····· 159

21.1 XrtGaTick ····· 159
21.2 Tick オブジェクトのリソースについて ····· 160
21.3 サンプル・コード ····· 162
21.4 リソースの一覧 ····· 162
21.5 手続きと方法 ····· 165

22 ユーザ・インタラクション ····· 167

22.1 XRT/gauge のユーザ・インタラクション ····· 167
22.2 トランスレーションとアクションの概要 ····· 167
22.3 Widget/Object インタラクションの開始 ····· 168

22.4 Widget/Object インタラクションのアップデート ····· 169
22.4.1 Widget/Object インタラクションの終了 ····· 170
22.5 移動とズーム ····· 170
22.6 トランスレーションとズームのアップデート ····· 172
22.6.1 トランスレーションとズーム ····· 172
22.6.2 トランスレーションとズーム ····· 173
22.7 Counter オブジェクトのリセット・アクション ····· 173
22.7.1 マッピング ····· 174
22.7.2 値変更のコールバック ····· 178
22.7.3 イベント・ハンドリングの手続き ····· 178
22.8 ウィンドウのリサイズ ····· 179

23 さらに進んだ使用法 ····· 181

23.1 独自の描画ファンクション ····· 181
23.1.1 XrtGaDrawData 構造体のパラメータ ····· 184
23.2 XtConvertAndStore でイメージをセットする ····· 186

24 PDS ユーティリティ ····· 189

24.1 XrtPDSImage ····· 189
24.1.1 XRTPDSImage の作成 ····· 189
24.2 Image 型の登録 ····· 190
24.2.1 定義済みのイメージ ····· 190
24.2.2 任意のイメージ型 ····· 191
24.2.3 イメージ・ファンクションを書く ····· 192
24.3 ダイナミックまたは共用ライブラリの使用 ····· 193
24.4 XrtPDSList オブジェクト ····· 193
24.4.1 サンプル・コード ····· 194
24.4.2 リストの作成と破棄 ····· 194
24.4.3 リストのアイテムの追加と削除 ····· 194
24.4.4 アイテムの検索と取得 ····· 196
24.4.5 リスト比較ルーチン ····· 199
24.4.6 リストのソート ····· 200
24.4.7 ユーザが定義したデータ ····· 201
24.5 手続きと方法 ····· 201
24.5.1 XrtPDSImage の方法 ····· 201
24.5.2 XrtPDSList オブジェクトの方法 ····· 203

PART II Appendices

A マクロと方法 ····· 215

A.1 マクロ ····· 215
A.2 方法 ····· 215

B Data 型 ····· 219

B.1 列挙法 ····· 219
B.2 構造体 ····· 224

C リソースの型とクラス ····· 227

C.1 Center オブジェクト ····· 227
C.2 Circular Dial ウィジェット ····· 227
C.3 Circular Gauge ウィジェット ····· 228
C.4 Circular Scale オブジェクト ····· 229
C.5 Counter オブジェクト ····· 230
C.6 Frame オブジェクト ····· 231
C.7 Icon オブジェクト ····· 231
C.8 Indicator オブジェクト ····· 231
C.9 Item オブジェクト ····· 232
C.10 LEDIcon ウィジェット ····· 232
C.11 LEDNumerical ウィジェット ····· 233
C.12 Linear Dial ウィジェット ····· 234
C.13 Linear Gauge ウィジェット ····· 235
C.14 Linear Scale オブジェクト ····· 236
C.15 Marker オブジェクト ····· 236
C.16 Needle オブジェクト ····· 237
C.17 Percent ウィジェット ····· 237
C.18 Range オブジェクト ····· 238
C.19 State オブジェクト ····· 238
C.20 Switch オブジェクト ····· 239
C.21 Text オブジェクト ····· 240
C.22 Tick オブジェクト ····· 241

D 継承されたリソース ····· 243

E リソース・ファイルと GUI ビルダ ····· 245

E.1 構文 ····· 245
E.2 サンプル・コードとリソース・ファイル ····· 247
E.2.1 サンプル・プログラム ····· 247

F UIL 引数 ····· 249

F.1 Center オブジェクトのリソース ····· 249

F.2 Circular Dial ウィジェットのリソース ····· 250
F.3 Circular Gauge ウィジェットのリソース ····· 250
F.4 Circular Scale オブジェクトのリソース ····· 251
F.5 Counter オブジェクトのリソース ····· 252
F.6 Frame オブジェクトのリソース ····· 253
F.7 Icon オブジェクトのリソース ····· 253
F.8 Indicator オブジェクトのリソース ····· 253
F.9 Item オブジェクトのリソース ····· 254
F.10 LEDIcon ウィジェットのリソース ····· 254
F.11 LEDNumerical ウィジェットのリソース ····· 256
F.12 Linear Dial ウィジェットのリソース ····· 257
F.13 Linear Gauge ウィジェットのリソース ····· 257
F.14 Linear Scale オブジェクトのリソース ····· 258
F.15 Marker オブジェクトのリソース ····· 258
F.16 Needle オブジェクトのリソース ····· 259
F.17 Percent ウィジェットのリソース ····· 259
F.18 Range オブジェクトのリソース ····· 260
F.19 State オブジェクトのリソース ····· 260
F.20 Switch オブジェクト ····· 261
F.21 Text オブジェクトリソース ····· 261
F.22 Tick オブジェクトのリソース ····· 262
F.23 短くなっている定数 ····· 263

G サンプル･コード ····· 265

G.1 例 ····· 265
G.2 デモ ····· 266

索引 ····· 267

はじめに

XRT/gauge について ■ 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/gauge について

XRT/gaugeウィジェット・ファミリ

XRT/gauge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追加します。このXRT/gaugeは、次のウィジェットで構成されています。

- *Circular Gauge* と *Circular Dial* は、円形の計器に値を表示します。Circular Gaugeでは、プログラムやマウス・ドラッグによって移動するポインタが数値を継続的に表示し、Circular Dialでは、Scale上の決められた位置にエンドユーザがポインタを配置することができます。
- *Linear Gauge* と *Linear Dial* は、上記のCircular GaugeやDialと同じ機能を持ちますが、Scaleは円形ではなく矩形です。
- *LED Numerical* は、時計やタイマ、カウンタ等のように数値をデジタル表示します。
- *LED Icon* は、グラフィック・イコライザのような表示をします。
- *Percent* は、割合を視覚的に表示します。

XRT/gaugeウィジェット・クラスは、その表示や動作をコントロールするリソースを持っています。XRT/gaugeを使用したプログラミングは、ウィジェットが一つ多いことを除けば、他のMotifのプログラミングと全く替わりません。

XRT/gauge オブジェクト

XRT/gaugeは、オブジェクト指向に基づいて設計されていますので、ウィジェットのリソースを何度でも再利用することができます。APIは非常にシンプルで簡単になっていますので、短時間でXRT/gaugeプログラミングを習得することができます。尚、オブジェクトに関しては、次の章で解説されています。

XRT/gaugeには、ユーザがマウスを使用してダイアルやゲージとインタラクションを行うことができるように、*User Interaction Callback*も含まれています。

前提知識

このマニュアルは、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書かれています。Motif や XRT/gauge 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 *の配列"、"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'の項(P3)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/gauge 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)

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/gauge 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とても役に立ちます。

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/gauge 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

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お問い合わせください。

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/index.htm
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/

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/gauge 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/gauge

1

XRT/gauge について

XRT/gauge とは ■ XRT/gauge と Motif のクラス階層
環境変数 ■ プログラムのコンパイルと許可
リソースファイルと GUI ビルダ ■ XRT/gauge で C++ を使用する
XRT/gauge の国際化 ■ XRT/gauge の共通なエレメント

1.1 XRT/gauge とは

XRT/gauge は、アナログやデジタルの表示を行うウィジェット集です。各 Gauge には、それに対応する Dial があります。どのウィジェットも、他の XRT 製品と同様に全て Motif ウィジェット・セットの拡張です。

XRT/gauge ウィジェット

XRT/gauge には、次のウィジェットが揃っています。

- *Circular Gauge* と *Circular Dial* - 円形の計測器。Switch や Needle が付属している。
- *Linear Gauge* と *Linear Dial* – 矩形の計測器。
- *Numeric LED* – 七つの部分からなる数字を使用して数値を保管、表示。
- *Icon LED* – アイコン・オブジェクトをオン、オフにして値を表示。
- *Percent* – 円または棒グラフに割合を表示。

次ページには、XRT/gauge のオブジェクトの簡単な解説があります。各オブジェクトは、章毎に詳しく解説されていますので、それらのリソースを簡単に見つけることができるようになっています。

オブジェクト

XRT/gauge は、オブジェクト指向に基づいて設計されています。各ウィジェットはオブジェクトで構成されており、これらのオブジェクトは複数のウィジェットで共通です。また、多くのリソースは複数のオブジェクトで共有されており、プログラミングが簡単に習得できるようになっています。オブジェクトを作成し、リソースをセットした後、同じ親ウィジェットを持つ他のオブジェクトと関連付けることができます。アプリケーションが実行されているときに変更しなければならない値は、*XtSetValues()* または *XtVaSetValues()* を呼び出すことにより変更することができます。

- *Center* オブジェクトは、Circular Gauge ウィジェットに属し、Scale の中央を示す役割を持ちます。
- *Circular Scale* は、Circular Dial と Circular Gauge の基本となるオブジェクトです。Scale の開始と終了の値や角度の範囲を指定し、関連するほかのオブジェクトの位置やサイズをコントロールします。
- *Counter* オブジェクトは、Circular Gauge と Circular Dial に使用され、odometer(走行距離計)のように数値を表示します。Counter の数値が増加がするにつれ、数字が置き換わるのではなくロールオバーするように設定することもできます。オブジェクトなので、Ciruclar Gauge や Circular Dial に配置して、同じデータを同時にアナログ、デジタルの両方で表示したり、別々の情報を表示させたりすることができます。
- *Frame* オブジェクトは、Circular Gauge、Circular Dial、Linear Gauge、Linear Dial、Percent ウィジェットに配置し、これらの内部のカラーパネルの役割をします。
- *Icon* オブジェクトは、LEDIcon ウィジェットに使用されるエレメントです。指定されたオンとオフの色を操作して、あるアクションの進行状況や値の変化を表示します。
- *Indicator* オブジェクトは、Linear Gauge や Linear Dial の一部として使用されます。Linear Gauge では、Indicator が Scale 上での値をマークします。Linear Dial では、一つの Indicator しか使用できません。
- *Item* オブジェクトは、Percent ウィジェットのエレメントです。
- *Linear Scale* オブジェクトは、Linear Gauge と Linear Dial の基本となるオブジェクトです。Linear Gauge では複数の Scale を使用することができますが、Linear Dial では、一つの Scale しか使用できません。
- *Marker* オブジェクトは、Circular Dial の一部で、現在選択されている状態を示します。
- *Needle* オブジェクトは、Circular Gauge 上で Scale 上の値を指します。Scale 上には複数の Needle、Circular Gauge には複数の Scale を表示させることができます。
- *Range* オブジェクトは、Circular Gauge、Circular Dial、Linear Gauge に使用され、Scale 上で指定された範囲を塗りつぶします。この機能は、例えば、温度計やタコメータの危険ゾーンを示す場合などに使用できます。

- *State* オブジェクトは、Circular Dial で、Switch が選択できる各状態を示します。
- *Switch* オブジェクトは、Circular Dial では Needle のような役割をしますが、ある一つの値だけしか選択できない非連続モードに使用されます。
- *Text* オブジェクトは、Circular Gauge、Circular Dial、Linear Gauge、Linear Dial、Percent ウィジェットに使用され、ラベルの役割をします。
- *Tick* オブジェクトは、Circular Gauge、Circular Dial、Linear Gauge、Linear Dial に使用され、アナログの目盛りの役割をします。

1.2 XRT/gauge と Motif のクラス階層

XRT/gauge 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、他の標準のウィジェットと同様に 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。Motif XmPushButton ウィジェットは、Xt Core ウィジェットのサブクラスの Xm Primitive のサブクラスの XmLabel のサブクラスです。

XRT/gauge も同じ階層構造にあてはめることができ、XRT/gauge の全てのコンポーネントは、Core、XmPrimitive から派生しています。従って、XRT/gauge の各ウィジェットでは、その親(Core や XmPrimitive)のウィジェットのクラスで定義されているリソースも使用できます。

1.3 環境変数

プログラミングを開始する前に今一度使用する環境をご確認ください。*XRTHOME*という環境変数が、XRT/gauge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、XRTのリリース&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/gauge 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、実行できることも必ずご確認ください。

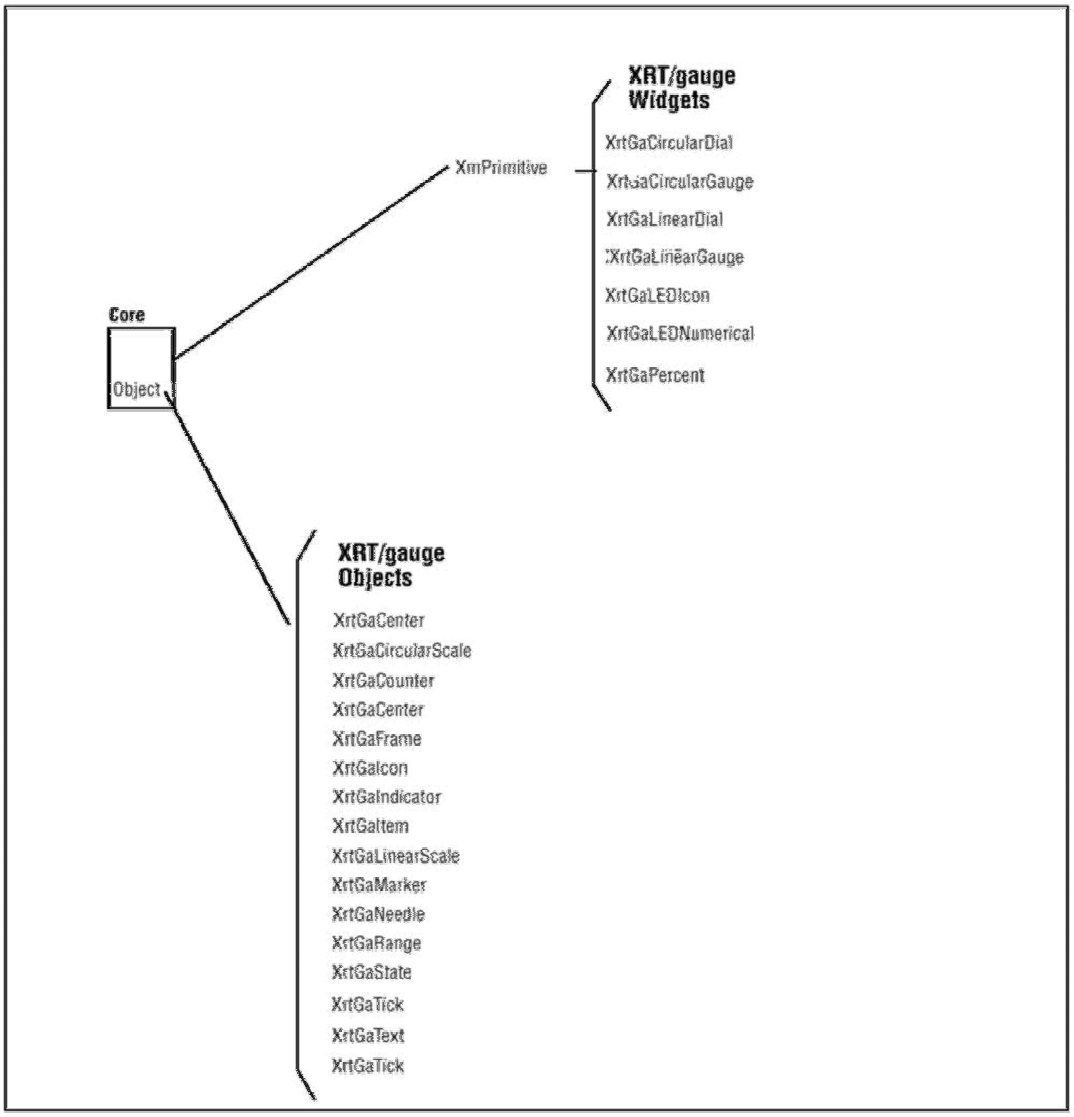


図1　Motif 階層内での XRT/gauge ウィジェットとオブジェクトの位置

1.4 プログラムのコンパイルと許可

X と Motif ライブラリ、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、次のようにして、*simple.c* をコンパイルし、プログラムに許可を与えることができます。[1]

```
cc simple.c -I$(XRTHOME)/include \
  -L$(XRTHOME)/lib -o simple \
  -lxrtgauge -lpdsutil-lXm -lXpm -l Xt -X11 -lm
$(XRTHOME)/bin/xrt_auth simple
```

xrt_auth

XRT/gauge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/gauge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 -qv executable_filename
```

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/gauge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ていないと、XRT/gaugeウィジェットは表示されますが、次のメッセージが*stdout*に出力されます。

```
This application is built with a disabled or an expired
time-limited version of XRT/gauge for Motif. Please contact
KL Group Inc. or your XRT/gauge for Motif reseller for
further information or to purchase a permanent license.
If you have a license, please check:
  1) That you have installed your license using the program
  $XRTHOME/bin/xrtpasswd or with the install script.
  2) That you have authorized your executable by running
  "xrt_auth". (For more information refer to
  "Compiling and Authorizing a Program" in Chapter 1 of the
  XRT/gauge for Motif Programmer's Guide & Reference Manual.)
  3) That you have the XRTHOME environment variabledefined.
  4) That you have permission to read the license file in
  $XRTHOME/etc/licenses.
  5) If you have a temporary license, make sure that it has
  not expired.
```

図2　許可が与えられていないか期限切れのアプリケーションによるメッセージ

注意:　共有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。

[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/gauge/examples*の*Makefile*にありますので参照してください。

C++と ANSI C

XRT/gauge の include ファイルは、C++と ANSI C、"Kernighan and Ritchie"C との互換性があります。

1.5 リソースファイルと GUI ビルダ

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ます。
- 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ます。

XRT/gauge のほとんどのリソースはリソースファイルで設定ができます。リソースファイルを使用することにより、異なるリソース設定のプロトタイプを簡単に作成したり、その結果を直ぐに見ることができます。P245 にある Appendix には、リソースファイルでのリソース設定の構文の解説がありますので参照してください。

GUI ビルダの使用

XRT/gauge は、多くの主要な GUI ビルダと統合して使用することができます。GUI ビルダから XRT/gauge のリソースをセットするには、リソースファイルの場合と同じ構文を使用してください。詳細は、P245 にある Appendix E を参照してください。

1.6 XRT/gauge で C++を使用する

XRT/gauge は、XRT/gauge ウィジェットに対する C++インターフェースを持っています。C++を使用するには、*XrtGauge.hxx* ファイルを 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、次の表のように各 XRT/gauge ウィジェットのクラスとコンストラクタを定義します。

Widget	Class	Constructor Function
Circular Dial	XrtGaCircularDial	XrtGaCircularDial()
Circular Gauge	XrtGaCircularGauge	XrtGaCircularGauge()
Linear Dial	XrtGaLinearDial	XrtGaLinearDial()
Linear Gauge	XrtGaLinearGauge	XrtGaLinearGauge()
LED Icon	XrtGaLEDIcon	XrtGaLEDIcon()
LED Numerical	XrtGaLEDNumerical	XrtGaLEDNumerical()
Percent	XrtGaPercent	XrtGaPercent()

次の表は、XRT/gauge ウィジェットに使用されるオブジェクトのクラスとコンストラクタの一覧です。

Object	Class	Constructor Function
Center	XrtGaCenter	XrtGaCenter()
CircularScale	XrtGaCircularScale	XrtGaCircularScale()
Counter	XrtGaCounter	XrtGaCounter()
Frame	XrtGaFrame	XrtGaFrame()
Icon	XrtGaIcon	XrtGaIcon()
Indicator	XrtGaIndicator	XrtGaIndicator()
Item	XrtGaItem	XrtGaItem()
LinearScale	XrtGaLinearScale	XrtGaLinearScale()
Marker	XrtGaMarker	XrtGaMarker()
Needle	XrtGaNeedle	XrtGaNeedle()
Range	XrtGaRange	XrtGaRange()
State	XrtGaState	XrtGaState()
Switch	XrtGaSwitch	XrtGaSwitch()
Text	XrtGaText	XrtGaText()
Tick	XrtGaTick	XrtGaTick()

その他、*XrtGauge.hxx* ファイルには、次の内容が含まれています。

- *XtDestroyWidget()*を呼び出すデストラクタ
- 各リソースをセットするメソッド
  例:`XmNxrtGaBorderType` をセットするメソッドは *setBorderType()*
- 各リソースを取り出すメソッド
  例: `XmNxrtGaBorderType` の値を取得するメソッドは *getBorderType()*
- Xt ファンクションを実行するメソッド *addCallback(), destroy(), getDisplay(), getName(), getParent(), isManaged(), isRealized(), manageChild().*

C++の例は*$XRTHOME/src/gauge/examples/simple_cpp.cxx* にあります。

1.7 XRT/gauge の国際化

XRT/gauge ウィジェットには、ご使用の X サーバでサポートされている Xwindow フォントならどれでも使用できます。ISO Latin1 は、北米と西欧言語の ISO 標準文字です。

ISO Latin1 は、ASCII のスーパーセットですので、ISO Latin1 フォントを指定しても、ASCII アプリケーションには影響しません。X Window フォントのうち、名前の最後に iso8859-1 とあるものは ISO Latin1 フォントです。

2byte フォント

漢字などの 2byte フォントも、String ではなく `XmString` 型であれば使用することができます。

1.8 XRT/gauge の共通なエレメント

1.8.1 マクロ

多くの XRT/gauge の例では、色の定義や倍精度の値の設定の際に生じる入力を減らすために二つのマクロを使用しています。下記は、これらのマクロで、*$XRTHOME/src/tools/mtgauge.h* にあります。

```
#define COLOR_RES(res, color) XtVaTypedArg, res, XmRString, \
                              color, strlen(color) + 1
#define DVAL(value)               XrtGaDoubleToArgVal(value)
```

1.8.2 Fallback リソースの設定

リソース値の設定を fallback 配列のエレメントとして定義することができます。下記のコードは、`XmFontList` リソースによって使用されるフォントの設定と、そして Linear Dial で使用されるオンとオフのイメージの設定の例です。

```
static String fallback_resources[] = {
  "*.XmFontList:-*-helvetica-medium-r-normal--*-120-*-*-*-*-*-*",
  "*.XmXrtGaLinearDial.xrtGaOnImage:
       $XRTHOME/src/icons/disk_3D_blue.xpm",
  "*.XmXrtGaLinearDial.xrtGaOffImage:
       $XRTHOME/src/icons/disk_3D_blue2.xpm",
  NULL
};
```

1.8.3 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()* の呼び出しの直後に反映されます。再描画される前にウィジェットのリソースを幾つか変更したい場合は、`XmNxrtGaRepaint` を `False` にしてください。全てのリソース変更は、`XmNxrtGaRepaint` が `True` にセットされるまで適応されません。

1.8.4 デバッグ

XmNxrtGaDebug

このリソースは、*Xt error* データベースから警告やエラーメッセージを XtAppWarningMessage にプリントするかどうかを指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

1.8.5 テキスト・オブジェクトのセンタリング

ウィジェット内で Text オブジェクトをセンタリングするには、MtCenterText を使用してください。

```
MtCenterText(widget, textObject);
```

この場合、widget はオブジェクトの親ウィジェト、textObject は Text オブジェクトを指します。これは、*$XRTHOME/src/tools/mtgauge.c* にあります。

$XRTHOME/src/tools/mtgauge.c には、その他に幾つか便利なメソッドがあります。

2

Circular Gauge と Circular Dial ウィジェット

ウィジェットの解説 ■ Circular Gauge と Circular Dial のオブジェクト
トランスレーションとアクション ■ サンプル・コード
ウィジェットの概要 ■ リソースの一覧
手続きとマクロ

この章では、Circular Gauge と Circular Dial に共通する機能とその操作法を解説します。

2.1 ウィジェットの解説

Circular Gauge と Circular Dial は、値を表示し、それを Scale 上で操作することができるインタラクティブな円形メータのコンテナです。これらは各種の機能を持ったオブジェクトで構成されており、インタラクティブなメータとして表示させたり、これらの機能をオフにして単純な表示ツールとして使用することができます。ウィジェットの背景は、範囲内を一色に塗りぶしたり、*frame* を使用して二色にしたり、イメージを表示することができます。

これらのウィジェットは、アナログ・デバイスとしても使用できます。より正確な数値をデジタル表示する場合には、Counter オブジェクトを配置することができます。次ページにある図 3 は、スピードメータとして使用されている Circular Gauge です。ここには、odometer(走行距離計)のように Counter を追加することもできます。

それぞれのオブジェクトが描画される順番や、イメージをダブル・バッファリングするかを指定することも可能です。

複数のインスタンスのオブジェクトが必要な場合、ウィジェットは、オブジェクト・グループ一つに対して一つのリストを保持します。Text オブジェクトを記録する Text リスト、(Scale 上の範囲を示す)*Range* と呼ばれるバンドの記録のリスト、Tick オブジェクトを記録する Tick リスト等がこれにあたります。

また、Circular Gauge が複数の Circular Scale オブジェクトを持つ場合、それら全てを記録するリストも持つことができます。

デフォルトでは、ウィジェットはレンダリングの順序を独自に決定しますが、描画される順にオブジェクトが記述されたリストを使用して、これを変更することができます。

Circular Gauge や Circular Dial はオブジェクトを組み合わせて構成されますが、Circular Scale は、それらのオブジェクト・グループに対するリファレンス・オブジェクトに当たります。各ウィジェットには、必ず少なくとも一つの Circular Scale が必要です。Circular Scale は、配置やサイズ、最小値・最大値、開始と終了の角度などをコントロールするリソースを持つオブジェクトです。また、Needle オブジェクトは Scale 上で値を指すオブジェクトです。この Needle は、Circular Scale の半径と原点を参照して、その中心と長さを決定します。長さは、Circular Scale の半径に対する割合から計算され、Circular Scale の位置やサイズが変更されれば、円の正しい中央の位置と Scale に対する正しい長さを保つために、Needle もそれに合わせて変更されます。Circular Scale では、背景色を指定することができますが、その他の視覚的情報は変更することができません。

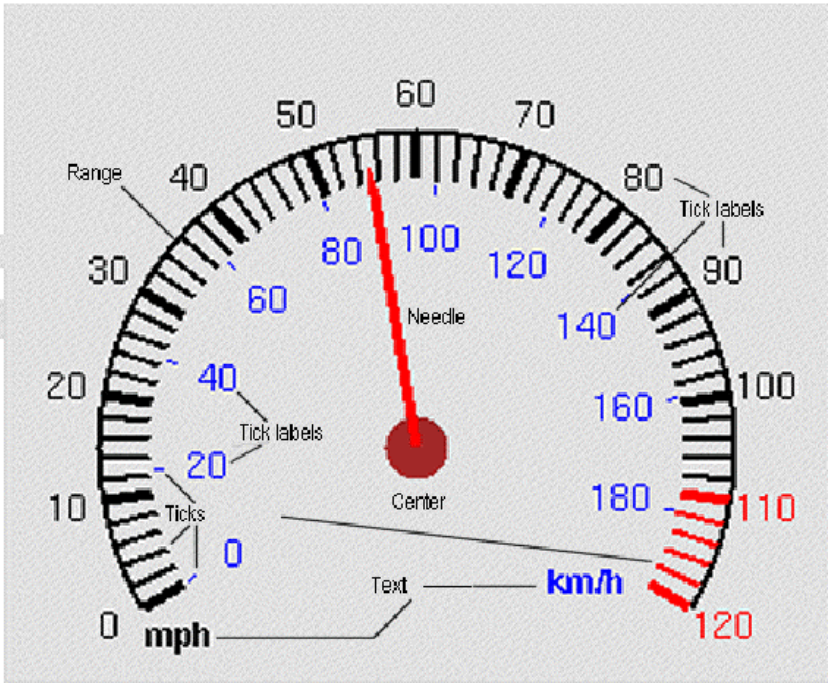


図3　Circular Scale とその他のオブジェクトを使用した Circular Gauge

Circular Dial は、Switch オブジェクトを持ち、非連続(discrete、値が Gauge のように連続していない)モードで表示されます。この非連続モードでは、選択できる値のリストが定義されています。アクティブな値(State)をハイライトしたり、その State とラベルを結ぶ線等のリソースが揃っています。

2.1.1　Circular Gauge と Circular Dial に共通なエレメント

Circular Gauge と Circular Dial には、共通なエレメントがあります。

背景色

この色はウィジェット自身に適用されます。セットされていない場合は、親から継承されます。

Circular Scale

Counter オブジェクトを除く、Gauge と Dial の視覚的なエレメントの基礎となるオブジェクトです。

Range オブジェクト

Scale 上の視覚的な指標または、Scale の一部のハイライトとして使用されます。複数の異なる色の Range オブジェクトを用いて、一つの Circular Scale を複数の分布に区切って表示することもできます。

Radius(半径)

Radius のパラメータは、Scale のサイズをセットします。

min、max(最小値と最大値)

これらの値は、Scale の開始と終了の値を指定します。

進行方向

Scale の値が増加する方向を指定し、時計周りまたは半時計回りに指定します。

開始と終了の角度

これらの値は、*min* と *max* の位置を指定します。Circular Scale は、0 度は真東に配置し、角度を半時計回りの方向に増加させます。開始角度を終了角度より大きくすることもでき、例えば、*min* 値は 12 時の位置、max 値は 3 時の位置とすることができます。この場合、開始角度は 90 度、終了角度は 0 度になります。

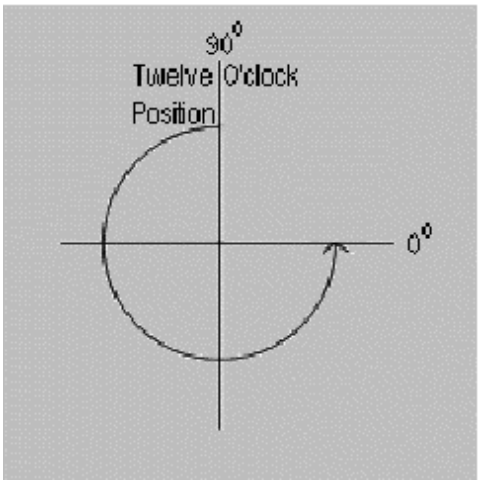


図4　Circular Scale オブジェクトの角度の見方

Tick

Tick オブジェクトは、Circular Scale 上に目盛りを表示します。

Frame

Circular Gauge 上の一部を異なる色でハイライトする場合は Frame を使用します。この Frame の場所は、ウィジェットの内側に描画されるハイライトの幅から計算されるマージンに依存します。

Render リスト

Render リストは、描画する子オブジェクトの順番を指定するリストです。通常、最初に Center オブジェクト、次に Needle が描画され、Needle が Center オブジェクトに隠されないようにします。このリストを変更して、描画されるオブジェクトの順序を変更することもできます。例えば、Needle を最初に描画して、円の中心ではなく、Center オブジェクトの端にこの針が付いているようにみせることができます。

2.1.2 Circular Gauge

Circular Gauge ウィジェットは、一つまたは複数の Circular Scale と Counter のコンテナです。その他に、下記のオブジェクトを使用することもできます。

- **Needles** - 現在値を示すポインタ。Circular Scale は、複数の Needle を持つことが可能
- **Center** - Circular Scale の中央をマークするグラフィカルなオブジェクト

2.1.3 Circular Dial

Circular Dial は、それぞれ独立した State を切り替えて値を表示します。非連続モードの場合、Dial が示す値は、どのアクションを開始するかを示す `case` ステートメントに使用することもできます。また、Dial は、Circular Gauge のように連続した値を持つように、連続モードにセットすることもできます。

- **Switch** – Circular Gauge が Needle を持つのに対し、Circular Dial は *Switch* を持ちます。一つの Circular Dial に対して一つのアクティブな Switch しか使用できません。新しい Switch を作成すると、Switch オブジェクトのリストに追加されます。一度に、このリストの Switch オブジェクトを二つ以上アクティブにすることはできません。Circular Dial の `XmNxrtGaSwitch` リソースが、アクティブな Switch を指定します。もし、アクティブな Switch がひとつしかなく、それが破棄されれば、このリソースは `NULL` にセットされます。
- **State** – State オブジェクトは、線やテキストを使用して Switch の位置を表示します。非連続モードでは、State リストが、Switch が取ることができる位置の数を決定します。連続モードでは、State は単なる Scale の注釈として表示されます。State のラベルにはテキスト・ストリングが使用されます。このオブジェクトは、*line inner radius*、*line outer radias*, *offset* という配置を指定する属性を持ちます。
- **Marker** – *Marker* は Switch にセットされ、Switch の位置を示します。一般的な使用法として、エンドユーザがマウス・クリックによって Switch の位置を指定し、その後、ウィジェットがこの Marker の位置をアップデートしていく方法があります。

2.2 Circular Gauge と Circular Dial のオブジェクト

2.2.1 Center オブジェクト(Circular Gauge)

Center オブジェクト(hub)は、Scale にセットされます。一つの Scale は、複数の Center を持つことができます。このオブジェクトでは、Radius と色を設定できるほか、イメージを表示させることができます。

2.2.2 Circular Scale(Circular Gauge または Circular Dial)

Circular Scale は、ユーザがコントロールできる原点の位置、開始と終了の角度、方向、最小と最大値を持っています。

一つの Circular Gauge 内には複数の Scale を使用することができます。Circular Scale をもう一つの Circular Scale の内側に配置して同じ原点を共有したり、または、それぞれを全く独立させることもできます。一つの Circular Dial は複数の Circular Scale を持つことができますが、一度には一つしかアクティブにすることはできません。このアクティブな Circular Scale は、Dial の `XmNxrtGaCircularScale` リソースが指しています。

Circular Scale での角度の測り方

Circular Dial と Gauge ウィジェットでの角度の測定方法は、一般的な幾何学の場合と同様に、真東を指す水平線から開始され、反時計方向に増加していきます。従って、真上を指す Needle は 90 度、真下では 270 度となります。

このデフォルトの方向を逆にすることもできます。デフォルトでは、Scale の *min* 値は開始角度にセットされますが、これを逆にすることができます。

XRT/gauge での、角度の測定法は、第 8 章の Circular Scale オブジェクトの章を参照してください。

Circular Gauge 内の Circular Scale の原点を変更する

下記の例では、Circular Gauge ウィジェットにある五つの Circular Scale に共通な原点をセットしています。ここでは、`XmNxrtGaXOrigin` と `XmNxrtGaYOrigin` が原点の変更に使用されています。

```
SetCircularGaugeOrigin(cgauge, xposition, yposition)
Widget cgauge; /* cgauge must be a Circular Gauge Widget */
 Position xposition;
 Position yposition;
 {
     XrtGaList    scale_list;
     Widget       scale_item[5];
     int          num_item;
```

```
int       i;
XtVaGetValues(cgauge,
      XmNxrtGaCircularScaleList,   &scale_list,
    NULL);

num_item = XrtPDSListGetItemCount(scale_list);

for (i=0; i<num_item; i++) {
 scale_item[i] = *(Widget *) XrtPDSListGetItem(scale_list, i);
}
scale_item[i] = NULL;

for (i=0; i<num_item; i++) {
 if (scale_item[i] != NULL) {
     XtVaSetValues(scale_item[i],
         XmNxrtGaXOrigin,     xposition,
         XmNxrtGaYOrigin,     yposition,
         NULL);
     }
 }
}
```

2.2.3 Frame

Circular Gauge と Circular Dial に Frame を指定して、ウィジェット内で異なった色の範囲を設定することができます。

Frame の形は矩形か円形でマージンをコントロールすることができます。マージンの値によっては、円形の Frame は楕円になることがあります。Frame は複数作成でき、ウィジェットはこれらのリストを持ちますが、一回に一つの Frame しかアクティブにはできません。

2.2.4 Marker(Dial)

Marker は、Circular Dial において Switch の現在の位置を示します。Marker は、Circular Scale の値を使用してその位置を決めます。Switch が左右対称な場合、選択された値は Marker よってのみ示されることになります。

2.2.5 Needle(Circular Gauge)

Needle は、Circular Scale 内で値を示すオブジェクトです。一つの Scale は複数の Needle を持つことができます。

Needle の色、長さ、幅、形を指定することができます。形は、*length* と *width* のパラメータによりそのサイズを変更します。

Needle は、*shape* という属性を使用して描画されます。三角、ポインタ、矢印などの他、独自の形を使用することができます。

Needle が示す、Scale 上の位置は、プログラムによってコントロールされるほか、エンドユーザのアクションによっても変更させることができます。

2.2.6 Range

Circular Scale においては、Range は、Scale 上の特定な場所を色によって強調する輪または部分的な輪です。

一つの Scale は複数の Range を持つことができます。

Range の内側と外側の半径をコントロールして幅を変更したり、開始と終了の値や色も指定することができます。

2.2.7 Switch

Switch は、通常、*オン*と*オフ*の二つの State を示しますが、連続していない複数の State を持つこともできます。連続モードの Switch は、*multi-turn-enabled* になり、最小値から最大値へ一回転(またはそれ以上)回転させることができます。

一つの Circular Dial は複数の Switch を持つことができ、これらを内部リストとして管理しますが、一度に一つの Switch しかアクティブにすることはできません。`XmNxrtGaSwitch` リソースが指している Switch を破棄すると、このリソースは `NULL` にセットされます。

図 5 は、四つの State を持つ矢印型のスィッチです。

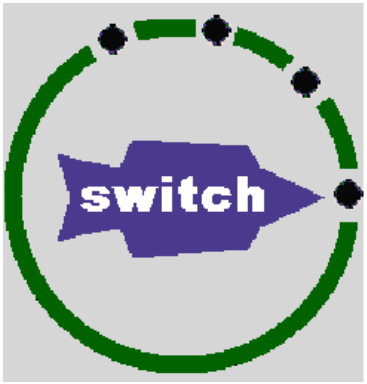


図5　四つの State を持つスィッチ・オブジェクト

2.2.8 State

State オブジェクトは、Circular Dial の一部です。非連続モードでは、この State リストが Switch が指すことができる位置の数を決定します。連続モードでは、State は Switch が指す場所を示します。非連続モードでは、各 State にラベルを付けることができます。ラベルの位置は、*line inner radius*、*line outer radius*、*offset* という属性によって決定されます。第 18 章　State オブジェクトの図 36 には、テキストの位置に関する詳細がありますので参照してください。

ユーザは、State オブジェクトのテキスト・ストリング、色、角度、配置、フォント、半径、半径からのオフセットなどをコントロールすることができます。

State は auto-aligned にして、開始と終了の角度の間に等分に自動配置させることができます。オフセットの線は、円形を描くようにしたり、または常に水平または垂直にすることができます。これらの半径や幅、表示・非表示などもユーザによってコントロールすることができます。

2.2.9 テキスト領域

Circular Gauge ウィジェットにはテキスト領域のリストを指定することができます。

この領域には、ユーザがコントロールできる、テキスト、位置、背景と前景色やイメージ、枠線などがあります。

このテキスト領域を Switch や Needle にセットし、*printf* と同じメカニズムを使用して、現在値を示すようにすることもできます。

2.2.10 Tick(Gauge)

Tick オブジェクトは、Scale 上に等分に配置された測定値をわかりやすく表示します。二種類の Tick を付けたい場合は、二つのオブジェクトを使用します。各オブジェクトには別々にプロパティをセットすることができますので、例えば、メインの目盛りにはラベル、サブの目盛りには何もつけないというようにセットすることができます。

Tick では、開始値と終了値、増分、長さのほか、色や幅をセットすることができます。

各 Tick オブジェクトに対するラベルの表示・非表示やフォントなども指定できます。

Tick の精度もセットすることができますが、精度を低くすることによって誤差が生じ、Scale が見にくくなることがありますので注意してください。

2.3 トランスレーションとアクション

Circular Gauge には、次のトランスレーションがあります。

```
<Btn1Down>:        ResetCounter(Begin) ActionStart(Needle)
<Btn1Motion>:      ActionMove(Needle)
<Btn1Up>:          ResetCounter(End) ActionEnd(Needle)
CTRL+<Btn1Down>:   ZoomStart(CircularGauge)
CTRL+<Btn1Motion>:ZoomUpdate(CircularGauge)
CTRL+<Btn1Up>:     ZoomEnd(CircularGauge)
SHIFT+<Btn1Down>: TranslateStart(CircularGauge)
SHIFT+<Btn1Motion>:TranslateUpdate(CircularGauge)
SHIFT+<Btn1Up>:    TranslateEnd(CircularGauge)
```

Circular Dial のトランスレーションは、次の通りです。

```
<Btn1Down>:       ResetCounter(Begin) ActionStart(Switch)
<Btn1Motion>:     ActionMove(Switch)
<Btn1Up>:         ResetCounter(End) ActionEnd(Switch)
CTRL+<Btn1Down>:  ZoomStart(CircularDial)
CTRL+<Btn1Motion>:ZoomUpdate(CircularDial)
CTRL+<Btn1Up>:    ZoomEnd(CircularDial)
SHIFT+<Btn1Down>: TranslateStart(CircularDial)
SHIFT+<Btn1Motion>: TranslateUpdate(CircularDial)
SHIFT+<Btn1Up>:   TranslateEnd(CircularDial)
```

2.4 サンプル・コード

下記のコードは、Circular Dial を作成し、そのプロパティをセットしています。

```
Widget cdial; /* The circular dial widget */
cdial =
     XtVaCreateManagedWidget("circular_dial",
             xmXrtGaCircularDialWidgetClass,  form,
             XmNleftAttachment,       XmATTACH_FORM,
             XmNrightAttachment,      XmATTACH_FORM,
             XmNtopAttachment,        XmATTACH_FORM,
             XmNxrtGaHighlightSelectedState,  True,
             XmNheight,                       350,
             XmNwidth,                        350,
             COLOR_RES(XmNxrtGaHighlightColor,"red"),
             COLOR_RES(XmNbackground,         "white"),
             NULL);
```

このウィジェットに Scale、Range、テキスト、State を追加し、図 6 の上部にあるような Dial を作成することができます。この図の下の部分にある三つの Circular Dial は、上部にある Dial のリソースをコントロールします。

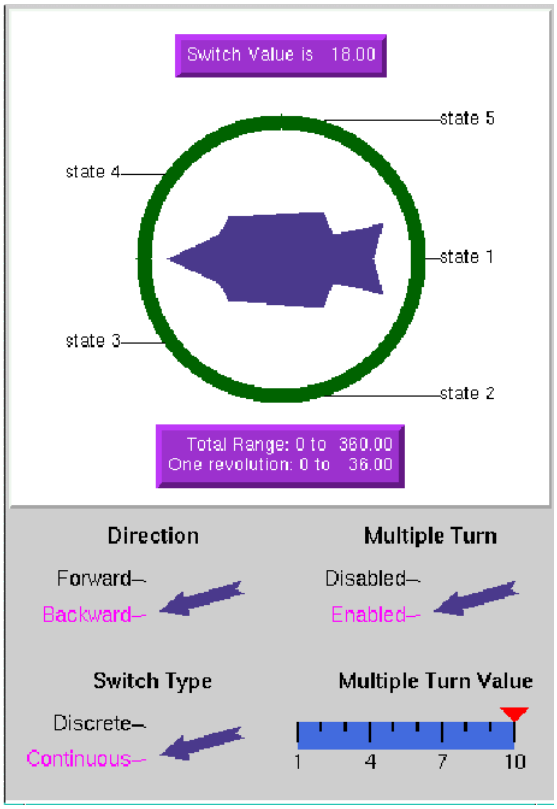


図6　サンプルの Circular Dial

Circular Dial で定義されているリソースの多くは、Circular Gauge でも定義することができます。下記のコードは、Circular Gauge を作成しています。

```
Widget cgauge; /* The Circular Gauge widget */
cgauge = XtVaCreateManagedWidget("cgauge",
     xmXrtGaCircularGaugeWidgetClass,form,
             XmNheight,           500,
             XmNwidth,            300,
             XmNleftAttachment,   XmATTACH_FORM,
             XmNtopAttachment,    XmATTACH_FORM,
             XmNbottomAttachment, XmATTACH_FORM,
             COLOR_RES(XmNbackground, "wheat"),
             NULL);
```

Frame オブジェクト、Text オブジェクト、Center オブジェクト、二つの State オブジェクト、Tick オブジェクト、Needle オブジェクトなどを追加し、このコードは、次のようなウィジェットを作成します。

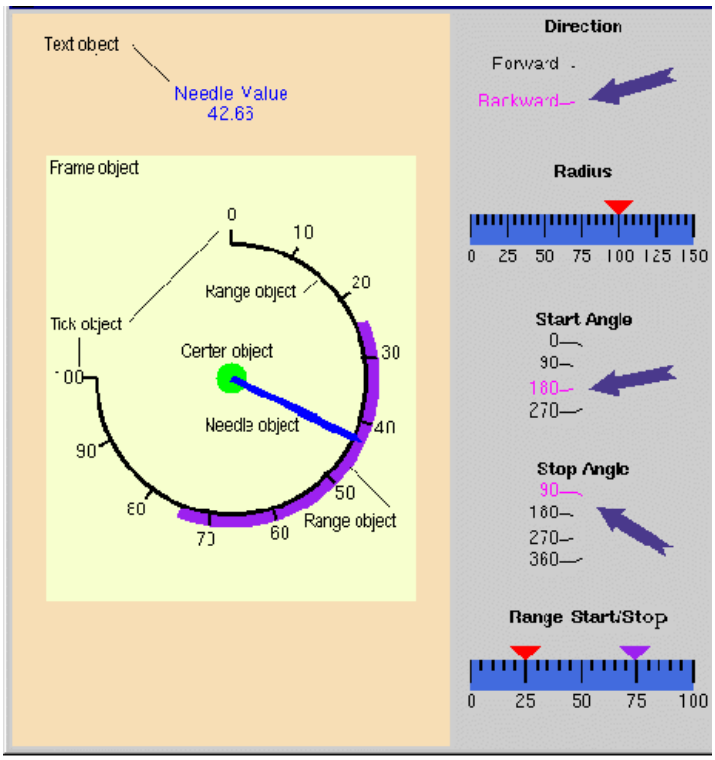


図7　サンプルの Circular Gauge

この図の右側には、Circular Dial と Linear Gauge が使用されています。

2.5　ウィジェットの概要

Circular Gauge

Include File:	*<Xm/XrtGaCircularGauge.h>*
Class Name:	XrtGaCircularGauge
Class Hierarchy:	Core ➔ XmPrimitive ➔ XrtGaCircularGauge
Class Pointer:	xmXrtGaCircularGaugeWidgetClass

Circular Dial

Include File:	*<Xm/XrtGaCircularDial.h>*
Class Name:	XrtGaCircularDial
Class Hierarchy:	Core ➔ XmPrimitive ➔ XrtGaCircularDial
Class Pointer:	xmXrtGaCircularDialWidgetClass

Circular Gauge と Circular Dial は、XmPrimitive のサブクラスなので、Circular Gauge は、*XtVaCreateManagedWidget()*によってコンストラクトすることができます。

Motif ウィジェットの階層や Circular Gauge と Circular Dial の階層内での位置に関しては、図 1 を参照してください。

2.6 リソースの一覧

Circualr Gauge と Circular Dial の全てのリソース及び、よく使用される継承されたリソースの一覧です。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。

2.6.1 両方の Circular ウィジェットに共通なリソース

XmNbackground **Pixel** ***inherited*** **CSG**

ウィジェットの背景色です。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_3D_OUT** **CSG**

ウィジェットの枠線の種類です。枠線の幅は `XmNshadowThickness` で指定します。

XmNxrtGaCounterList **XrtGaList** ***dynamic*** **G**

`XrtGaCounter` オブジェクトのリストです。新しい Counter オブジェクトが作成されると、親の Counter リストに追加されます。このリストは read only で操作することはできません。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースからの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換のエラーのみがプリントされます。

XmNxrtGaDoubleBuffer **Boolean** **True** **CSG**

ウィジェットの描画をダブル・バッファリングするかどうか指定します。ダブル・バッファリングは、オブジェクトの描画や再描画時に起きるちらつきをなくす機能です。

XmNxrtGaExposeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge の再描画をアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。`XrtGaCallbackStruct` のポインタが渡されます。

XmNxrtGaFrame **XrtGaObject** **NULL** **G**

ウィジェットの `XrtGaFrame` オブジェクトです。Frame の型は、`XmNxrtGaShape` によってコントロールされます。デフォルトでは、親ウィジェットに対して Frame オブジェクトは作成されません。作成された新しい Frame は、既に存在していた Frame と置き換わり、古い Frame は、ウィジェットによって Frame リストで管理されます。

このリソースが指しているFrameが破棄されると、このリソースはNULLにセットされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

ウィジェットの背景のイメージです。NULLの場合は背景色が使用されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージの位置を指定します。有効な値は XRTGA_IMAGE_LAYOUT_CENTERED と XRTGA_IMAGE_LAYOUT_TILED です。

XmNxrtGaInteractionCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge のインタラクションが起きたことをアプリケーションに通知するときに呼び出されるコールバックのリストを指定します。*ActionStart()*、*ActionMove()*、*ActionEnd()*、*RsetCounter()* の時に呼び出されます。XrtGaInteractionCallbackStruct のポインタが渡されます。尚、このコールバックの解説は第 22 章にもあります。

XmNxrtGaMapCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge の *Map()*アクションが呼び出されたことをアプリケーションに通知するときに呼び出されるコールバックのリストを指定します。XrtGaMapCallbackStruct のポインタが各 map コールバックに渡されます。*Map()*アクションを呼び出すデフォルトのトランスレーションはありませんが、登録することができます。

XmNxrtGaRangeList **XrtGaList** **NULL** **G**

XrtGaRange オブジェクトのリストです。(このオブジェクトの対象となる Scale は XmNxrtGaCircularScaleList のうちのひとつでなければなりません。)新しく作成された Range オブジェクトは、自動的の親の Range リストに追加されます。このリストは read only で内容を編集することはできません。

XmNxrtGaRenderList **XrtGaList** ***dynamic*** **G**

これと対のリソース、XmNxrtGaUseRenderList が True であれば、XmNxrtGaRenderList のリストに指定されている順序で子オブジェクトが描画されます。オブジェクトが作成されると、自動的に Render リストに追加されます。デフォルトでは、Render リストは子オブジェクトを作成された順序で持っています。このリソースは read only ですが、このリストのポインタを取得して直接編集することができます。

XmNxrtGaRepaint **Boolean** **True** **CSG**

リソースが変更された時にウィジェットを再描画するかどうかを決定します。ウィジェットに対して多くのリソース変更を加える場合は、XmNxrtGaRepaint を False にし、全ての変更が終了した時点で True に戻してください。

XmNxrtGaResizeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットのリサイズをアプリケーションに通知するときに呼び出されるコールバックのリストを指定します。XrtGaCallbackStruct のポインタが渡されます。

XmNxrtGaTextList **XrtGaList** ***dynamic*** **G**

XrtGaText オブジェクトのリストです。新しくテキスト・オブジェクトが作成されると自動的に親の Text リストに追加されます。このリストは、read only でその内容を編集することはできません。

XmNxrtGaTickList **XrtGaList** ***dynamic*** **G**

XrtGaTick オブジェクトのリストです。新しく作成された Tick オブジェクトは親の Tick リストに自動的に追加されます。このリストは read only でその内容を編集することはできません。

XmNxrtGaUseRenderList **Boolean** **False** **G**

このリソースの値が True の場合、ウィジェットは XmNxrtGaRenderList のリストに指定されている順序で子オブジェクトを描画します。このリソースが False の場合は、子オブジェクトはデフォルトの順序で描画されます。デフォルトでは各ウィジェットに適するような順序でオブジェクトが描画されるようになっていますので、ほとんどの場合、デフォルトのままにしておくことができます。

2.6.2 Circular Gauge – その他のリソース

XmNxrtGaCenterList **XrtGaList** ***dynamic*** **G**

XrtGaCenter オブジェクトのリストです。新しく作成された Center オブジェクトが自動的に親の Center リストに追加されます。このリストは read only でその内容を編集することはできません。

XmNxrtGaCircularScaleList **XrtGaList** ***dynamic*** **G**

XrtGaCircularScale オブジェクトのリストです。新しく作成された Circular Scale オブジェクトは自動的に親の Circular Scale リストに追加されます。このリストは read only でその内容を編集することはできません。

XmNxrtGaNeedleList **XrtGaList** ***dynamic*** **G**

XrtGaNeedle オブジェクトのリストです。新しく作成された Needle オブジェクトは自動的に親の Needle リストに追加されます。このリストは read only でその内容を編集することはできません。

2.6.3 Circular Dial – その他のリソース

XmNxrtGaCircularScale **XrtGaObject** **NULL** **G**

ウィジェットの Circular Scale です。ウィジェットは、Circular Scale の内部リストを管理します。新しく Circular Scale が作成されると古いものと置き換わります。指されているオブジェクトが破棄されるとこのリソースは NULL にセットされ、(counter が作成されていない限り) ウィジェットには何も描画されません。

XmNxrtGaHighlightColor **Pixel** ***inherited*** **CSG**

非連続モードの Switch で選択されている State をハイライトするときに使用される色です。デフォルトでは、親ウィジェットの前景色が使用されます。

XmNxrtGaHighlightSelectedState **Boolean** **False** **CSG**

このリソースが True の場合、非連続モードの Switch で選択されている State の線とテキストにハイライト色が使用されます。連続モードの Switch では、State より 2 度未満の角度であれば、ハイライト色が使用されます。

XmNxrtGaLineInnerRadius double [1] 1.0 CSG

(State の XmNxrtGaLineInnerRadiusUseDefault が True であれば)、Switch の各 State の線の内側の半径を決定します。この値は、Switch がセットされている Circular Scale の XmNxrtGaRadiusSize 値と関連します。例えば、0.5 という値は、Switch オブジェクトの内側の半径が Scale の半径の半分であることを意味します。

XmNxrtGaLineOuterRadius double [1] 1.0 CSG

(State の XmNxrtGaLineOuterRadiusUseDefault が True であれば)、Switch の各 State の線の外側の半径を決定します。この値は、Switch がセットされている Circular Scale の XmNxrtGaRadiusSize 値と関連します。例えば、0.5 という値は、Switch オブジェクトの外側の半径が Scale の半径の半分であることを意味します。

XmNxrtGaMarker XrtGaObject NULL G

Switch の現在値に表示される Marker を指定します。このリソースが NULL であれば、Marker は描画されません。デフォルトでは、Marker オブジェクトは作成されません。新しく Marker が作成されると、既存の Marker と置き換わります。値の内部リストがウィジェットによって管理されます。現在の Marker が破棄されると、このリソースは NULL にセットされます。

XmNxrtGaOffsetDistance double [1] 1.0 CSG

XmNxrtGaOffsetType が XRTGA_OFFSET_TYPE_HORIZONTAL または XRTGA_OFFSET_TYPE_VERTICAL で、オフセットが(State ではなく [2])ウィジェットによってコントロールされているのであれば、このオフセットの距離が使用されます。このリソースは、ウィジェットの中央を通る水平または垂直な線からの距離を決定します。各 State において、線は内径から外径を通り、この距離によって決定される位置にまで描画されます。このリソース値は、セットされている Circular Scale の XmNxrtGaRadiusSize に基づいています。例えば、1.5 という値は、距離が半径の 1.5 倍であることを意味します。尚、このリソースは、XmNxrtGaOffsetType が XRTGA_OFFSET_TYPE_CIRCULAR または XRTGA_OFFSET_AUTO の場合は、何もしません。

XmNxrtGaOffsetType XrtGaOffsetType XRTGA_OFFSET_TYPE_AUTOMATIC CSG

各 State では、(XmNxrtGaOffsetUseDefault が True であれば)ストリングのオフセットはウィジェットによって自動的にコントロールされます。このリソースはそのオフセットの種類を決定します。オフセットの線は自動的にセットしたり、円形のパターンを描いたり、常に水平または直角になるようにセットできます。有効な値は次の通りです。

XRTGA_OFFSET_TYPE_AUTOMATIC	デフォルトのパターン
XRTGA_OFFSET_TYPE_CIRCULAR	円形のパターン
XRTGA_OFFSET_TYPE_HORIZONTAL	水平のオフセット
XRTGA_OFFSET_TYPE_VERTICAL	垂直のオフセット

---

[1] リソースにダブル値をセットする場合は、常に *XrtGaDoubleToArgValue()* またはそれと同等のファンクション *DVAL()* を使用してください。詳細は、第 1 章のマクロの解説を参照してください。

[2] 連続モード、非連続モードどちらの場合も Switch の State は同じように描画されます。異なるのは Switch の動作です。

XmNxrtGaStateList **XrtGaList** ***dynamic*** **G**

このリストは、選択できる State を定義します。新しく State オブジェクトが作成されると親の State オブジェクトに自動的に追加されます。このリストは read only でその内容を操作することはできません。

XmNxrtGaSwitch **XrtGaObject** **NULL** **G**

ウィジェットによって使用される Switch です。このリソースが NULL であると、(ウィジェットが Counter を持っていない場合)背景色のみが表示されます。新しい Switch が作成されると、既存の Switch と置き換わります。現在の Switch が破棄されると、このリソースは NULL にセットされます。

2.7 手続きとマクロ

ここでは、Circular Gauge と Circular Dial の手続きを解説します。

XmCreateXrtGaCircularDial()

XrtGaCircularDial ウィジェットを作成する標準の Motif 形式のファンクションです。新しく作成された XrtGaCircularDial ウィジェットを返しますが、ウィジェットが作成されなければ NULL を返します。

```
Widget
XmCreateXrtGaCircularDial (
      Widget       parent,
      char         * name,
      Arg          * arglist,
      Cardinal     argcount)
```

parent は、作成される XrtGaCiruclardial ウィジェットの親、*name* は新しく作成された XrtGaCircularDial ウィジェットに与えられる名前です。*arglist* は、作成時に XrtGaCircularDial に提供される XrtArgVal のリソースと値のペアで、NULL とすることもできます。*argcount* は arglist 内のエレメントの数です。

XmIsXrtGaCircularDial()

```
XmIsXrtGaCircularDial(w)
```

w が XrtGaCircularDial または XrtGaCircularDial のサブクラスであれば、このマクロは True を返します。

XmCreateXrtGaCircularGauge()

XrtGaCircularGauge ウィジェットを作成する標準の Motif 形式のファンクションです。新しく作成された XrtGaCircularGauge ウィジェットを返しますが、ウィジェットが作成されなければ NULL を返します。

```
Widget
XmCreateXrtGaCircularGauge (
          Widget       parent,
          char         * name,
          Arg          * arglist,
          Cardinal     argcount)
```

parent は、作成される XrtGaCiruclarGauge ウィジェットの親、*name* は新しく作成された XrtGaCircularGauge ウィジェットに与えられる名前です。*arglist* は、作成

時に XrtGaCircularGauge に提供される XrtArgVal のリソースと値のペアで、NULL とすることもできます。*argcount* は arglist 内のエレメントの数です。

XmIsXrtGaCircularGauge()

```
XmIsXrtGaCircularGauge(w)
```

w が XrtGaCircularGauge または XrtGaCircularGauge のサブクラスであれば、このマクロは True を返します。

3

Linear Gauge と Linear Dial ウィジェット

ウィジェットについて ■ サンプル・プログラム
トランスレーションとアクション ■ ウィジェットの概要
リソースの一覧 ■ 手続きとマクロ

3.1 ウィジェットについて

Circular Gauge や Circular Dial と同様に、Linear Gauge と Linear Dial も数値を示す計器の役割を果たします。各 Gauge または Scale には、必ずひとつの Linear Scale があります。Linear Scale は、温度や長さ、量などの測定に適しています。

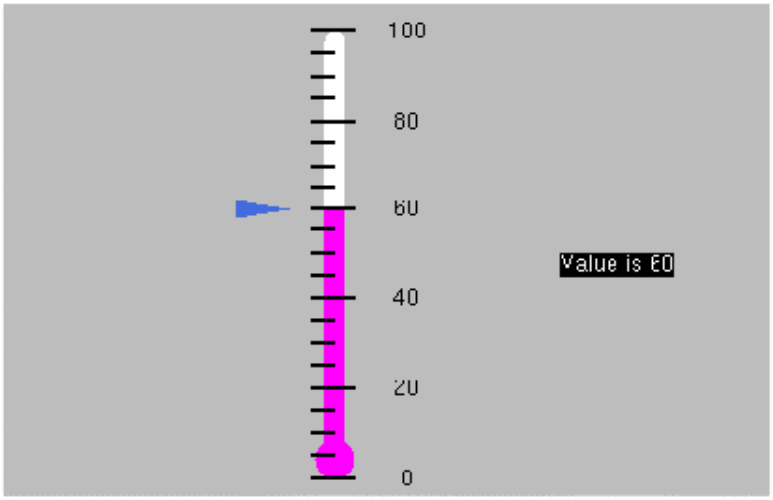


図8　基本的な Linear Dial

3.1.1 Linear Gauge と Linear Dial に共通なエレメント

背景と前景色

どちらのウィジェットにも背景色と前景色をセットすることができ、これらはそれぞれ XmNbackground と XmNforeground でコントロールできます。

ウィジェットの前景色は自動的に子オブジェクトの前景色となります。例えば、ウィジェットに Text オブジェクトを追加した時にその前景色を指定していなければ、テキストの色はその親のウィジェットから継承されます。また、前景色は、Switch などの幾つかのオブジェクトでは、背景色として継承されることもあります。

枠線

Linear Gauge と Dial ウィジェットは、XmNGaBorder という枠線のリソースを持っています。このリソースはウィジェットの周囲に描かれる枠線の種類をコントロールします。枠線の幅の設定には、XmNshadowThickness を使用してください。詳細は P41 を参照してください。

```
lgauge =
 XtVaCreateManagedWidget("lgauge",
     XmNshadowThickness,  25,
     XmNxrtGaBorder,      XRTGA_BORDER_ETCHED_OUT,
     NULL);
```

XmNxrtGaDebug

XmNxrtGaDebug を使用すると、ウィジェットに対するより詳しい診断メッセージがプリントされます。

ウィジェットの描画

XmNxrtGaUseRenderList が True にセットされていなければ、Linear ウィジェットはそれに適したデフォルトの順序で描画されていきます。オブジェクトは、ウィジェットに追加されていく順に Render リストに追加されます。リスト内のオブジェクトの順序は、そのまま描画される順序となりますので、このリストをデフォルトより変更し、XmNxrtGaUseRenderList を True にセットすれば、編集されたリストが描画の順序をコントロールするようにできます。

既存の Render リストを変更するには、次のようにしてください。

```
Boolean       okay;
XrtGaList     renderlist;
XtVaGetValues(lgauge,
          XmNxrtGaRenderList, &renderlist,
          NULL);
okay = XrtPDSListMoveItemsAfter(renderlist, 6, 3, 1);
```

*XrtPDSListMoveItemsAfter()*は、既存の render リストの順序を変更します。ここでは、item3 の後に item6 を移動しています。最後のパラメータは、一つのアイテムだけを移動することを示しています。

Expose コールバック

ウィジェットの再描画時に幾つかのタスクを実行するときには、XmNxrtGaExposeCallback コールバックを使用します。Linear Gauge または Dial ウィジェットと一緒に他のウィジェットを使用する場合、それらを Linear Gauge または Dial の上に配置されるようにする時は、Linear ウィジェットで隠されてしまわないように、そのウィジェットが必ず再描画されるようにしてください。

Linear Scale

Linear Scale は、Linear Gauge に測定機能を提供します。Linear Gauge には、複数の Scale を使用することができます。ウィジェットは、作成された Linear Scale のリストを管理します。

イメージの使用

ウィジェットの背景にはイメージを使用することができます。イメージは中央に配置したり、タイルのように並べたりすることができます。

Resize コールバック

Linear Gauge または Dial を持つウィンドウがリサイズされた後に、ウィジェットを再配置する必要がある場合は、XmNxrtGaResizeCallback を使用してください。

Text オブジェクトの配置

ウィジェット内のアイテムに注釈を付ける場合は、Text オブジェクトをラベルとして使用してください。Linear Gauge、Linear Dial のどちらにも複数の Text オブジェクトを貼り付けることができます。XmNxrtGaTextList は、作成された Text オブジェクトのリストを維持します。

Tick リスト

Tick オブジェクト(目盛り)は、スケールの値を読み取る時にガイドとなるマークです。Circular Gauge や Dial と同様に、Linear Gauge や Dial も二つ以上の Tick のセットを持つことができます。これらのオブジェクトは XmNxrtGaTickList に管理され、参照したり操作したりすることができます。

3.1.2 Linear Gauge オブジェクト

Indicator

Indicator は、スケール上の値を指すポインタです。この Indicator は、矩形や円形、または温度計のような形にしたり、(中央に配置されたり、タイルのように並べられた)イメージを持つこともできます。

Linear Gauge には複数の Indicator を使用することができますので、例えば、高低の値と現在値を同時に示すことができます。また、Range オブジェクトと合わせて使用し、Scale を読みやすくすることもできます。Linear Gauge ウィジェットは、Indicator オブジェクトが作成されると、read-only のリストにそれらを追加していきます。リスト上の Indicator は、XmNxrtGaIndicatorList リソースによってアクセスすることができます。

Indicator の位置は、値とその値がどのように Linear スケールに関係しているかによって決まります。

Range(範囲)

Range は、Linear スケール上に配置されるオブジェクトで、これのリストは、XmNxrtGaRangeList によってアクセスできます。

Linear Dial は Range リストを持ちませんが、その代わりに、Linear Dial には独自の形や*オン*と*オフ*の色を指定することができます。

再描画のバッチ処理

ウィジェットに多くの変更を行う場合は、再描画をコントロールしてください。XmNxrtGaRepaint を False にしてから、ウィジェットに変更を加え、その後、このリソースを True にしてください。この処理により、再描画をスムーズに行うことができます。

3.1.3 Linear Dial オブジェクト

Linear Dial オブジェクトは、一つの Linear Scale と一つの Indicator を持ちます。Dial は、Scale 上の*オン*と*オフ*の範囲を自動的に表示しますので、Gauge よりプログラミングを簡単に行うことができます。Scale は、値と*オン*と*オフ*の色またはイメージを提供するだけで自動的にアップデートします。Linear Gauge と同等の機能を実現するには、*オン*と*オフ*の二つの範囲を調節するように、コールバックを書く必要があります。

Indicator、オンとオフの色またはイメージ

Linear Dial の現在値を指す Indicator は、Circular Dial の Needle にあたります。この Indicator が指す値と同じかそれ以下の値の範囲を*オン*、それより上の値を*オフ*の範囲とし、こららの範囲には、それぞれの色またはイメージを指定して、Indicator が指す値をわかりやすく表示することができます。

XmNxrtGaIndicator は、現在アクティブになっている Indicator を指します。XmNxrtGaOffColor と XmNxrtGaOnColor リソースは*オン*と*オフ*の色、XmNxrtGaOffImage と XmNxrtGaOnImage は*オン*と*オフ*のイメージをコントロールします。

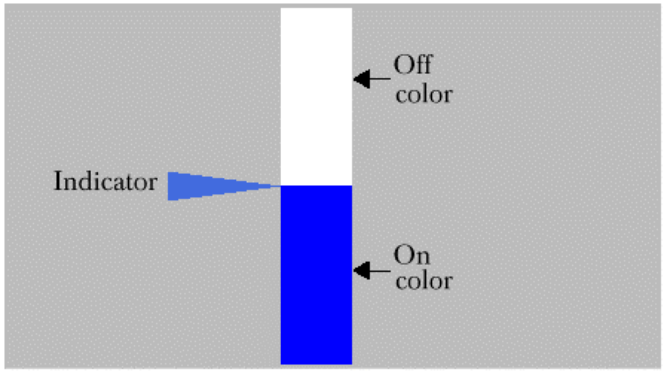


図9　オンとオフの色でその位置が強調されている Indicator

Linear Dial は、ウィジェットと一緒に作成されるデフォルトの Indicator を持っています。この Indicator は、新しく他の Indicator が作成されると破棄されます。このデフォルトの Indicator は XmNxrtGaIndicator リソースが指しています。デフォルトでは、XmNxrtGaIsShowing は False にセットされており、Indicator が表示されていなくても、*オン*と*オフ*の範囲が変更できるようになっています。

オンとオフの範囲には色の代わりにイメージを使用することができます。XmNxrtGaOnImage と XmNxrtGaOffImage を使用してイメージを指定してください。下記の例では、fallback リソース・ファイルでイメージをセットしています。

```
static String fallback_resources[] = {
      "*.XmXrtGaLinearGauge.xrtGaOnImage:
      $XRTHOME/src/icons/onImage.xpm",
      "*.XmXrtGaLinearGauge.xrtGaOffImage:
      $XRTHOME/src/icons/offImage.xpm",
      NULL
};
```

オンとオフの範囲の形

XmNxrtGaShape を使用して*オン*と*オフ*の範囲の形を指定できます。選択できる値は、XRTGA_SHAPE_RECTANGLE、XRTGA_SHAPE_CIRCLE、XRTGA_SHAPE_THERMOMETER、XRTGA_SHAPE_TANK です。

3.2 サンプル・プログラム

下記は、Linear Gauge を構築するプログラムの一部です。最初に、ウィジェットを持つ form が作成され、そこにウィジェットが追加されます。次に Linear Scale、Range、Indicator オブジェクトが追加され、それらのうちの幾つかのリソースがセットされています。値変更のイベントが起きた時に呼び出されるファンクションのリストにコールバックが追加されています。ここにないファンクションは、*$XRTHOME*/*src*/*gauge*/*examples*/*lgauge.c*.にあります。

```
Widget    form; /* The form that holds the widget */
Widget    lgauge, lg_scale, range, indicator;
/* Create Linear Gauge */
lgauge =
 XtVaCreateManagedWidget("lgauge",
      xmXrtGaLinearGaugeWidgetClass,   form,
      NULL);
/*Create Basic Scale*/
lg_scale = XtVaCreateWidget("lg_scale",
      xmXrtGaLinearScaleObjectClass,   lgauge,
      XmNxrtGaMax,         DVAL(200.0),
      XmNwidth,            250,
      XmNx,                25,
      XmNheight,           25,
      XmNy,                250,
      NULL);
/*Create Range*/
range = XtVaCreateWidget("range",
      xmXrtGaRangeObjectClass,lgauge,
```

```
        XmNxrtGaScale,           lg_scale,
        XmNxrtGaInnerExtent,     DVAL(0.0),
        XmNxrtGaOuterExtent,     DVAL(1.0),
        COLOR_RES(XmNbackground, "paleturquoise"),
        NULL);
    /*Create Indicator*/
    indicator_type = XRTGA_INDICATOR_TYPE_TRIANGLE;
    interaction_type = XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE;
    indicator =
     XtVaCreateWidget("indicator",
        xmXrtGaIndicatorObjectClass, lgauge,
        XmNxrtGaLinearScale,         lg_scale,
        XmNwidth,                    20,
        XmNxrtGaValue,               DVAL(0.0),
        XmNxrtGaIndicatorType,       indicator_type,
        XmNxrtGaInteractionType,     interaction_type,
        XmNxrtGaInnerExtent,         DVAL(-0.5),
        XmNxrtGaOuterExtent,         DVAL(0.0),
        COLOR_RES(XmNbackground,     "plum"),
        NULL);
    XtAddCallback(indicator, XmNxrtGaValueChangedCallback, lg_ValueCB,
     NULL);
```

3.3 トランスレーションとアクション

Linear Gauge のトランスレーションは次の通りです。

```
<Btn1Down>:       ActionStart(Needle)
<Btn1Motion>:     ActionMove(Needle)
<Btn1Up>:         ActionEnd(Needle)
CTRL+<Btn1Down>:  ZoomStart(LinearGauge)
CTRL+<Btn1Motion>: ZoomUpdate(LinearGauge)
CTRL+<Btn1Up>:    ZoomEnd(LinearGauge)
SHIFT+<Btn1Down>: TranslateStart(LinearGauge)
SHIFT+<Btn1Motion>:TranslateUpdate(LinearGauge)
SHIFT+<Btn1Up>:   TranslateEnd(LinearGauge)
```

Linear Dial のトランスレーションは次の通りです。

```
<Btn1Down>:       ActionStart(Switch)
<Btn1Motion>:     ActionMove(Switch)
<Btn1Up>:         ActionEnd(Switch)
CTRL+<Btn1Down>:  ZoomStart(LinearDial)
CTRL+<Btn1Motion>: ZoomUpdate(LinearDial)
CTRL+<Btn1Up>:    ZoomEnd(LinearDial)
SHIFT+<Btn1Down>: TranslateStart(LinearDial)
SHIFT+<Btn1Motion>: TranslateUpdate(LinearDial)
SHIFT+<Btn1Up>:   TranslateEnd(LinearDial)
```

3.4 ウィジェットの概要

Linear Gauge	Include File:	*<Xm/XrtGaLinearGauge.h>*
	Class Name:	XmXrtGaLinearGauge

	Class Hierarchy:	Core ➔ XmPrimitive ➔ XmXrtGaLinearGauge
	Class Pointer:	XmXrtGaLinearGaugeWidgetClass
Linear Dial	Include File:	*<Xm/XrtGaLinearDial.h>*
	Class Name:	XmXrtGaLinearDial
	Class Hierarchy:	Core ➔ XmPrimitive ➔ XmXrtGaLinearDial
	Class Pointer:	xmXrtGaLinearDialWidgetClass

Motif ウィジェットの階層や Linear Gauge と Linear Dial の階層内での位置に関しては、図 1 を参照してください。

3.5 リソースの一覧

Linear Gauge と Linear Dial の全てのリソース及び、よく使用される継承されたリソースの一覧です。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。

3.5.1 共通なリソース

XmNbackground **Pixel** ***inherited*** **CSG**

ウィジェットの背景色です。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_3D_OUT** **CSG**

ウィジェットの枠線の種類です。`XmNshadowThickness` が枠線の幅を指定します。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースからの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDoubleBuffer **Boolean** **True** **CSG**

ウィジェットの描画をバッチ処理するかどうか指定します。ダブル・バッファリングは、オブジェクトが描画または再描画するときのちらつきを現象されるテクニックです。

XmNxrtGaExposeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットがその一部を再描画したことをアプリケーションに通知するときに呼び出されるコールバックのリストを指定します。`XrtGaCallbackStruct` のポインタが渡されます。このコールバックを使用して、ウィジェットが他のウィジェット上に描画され、再描画の必要があるときに通知を受けることができます。

XmNxrtGaFrame **XrtGaObject** **NULL** **CG**

ウィジェットの XrtGaFrame オブジェクトです。Frame の種類は XmNxrtGaShape によってコントロールできます。詳細は P96 の 10.4 を参照してください。デフォルトでは、親ウィジェットには Frame オブジェクトは作成されません。新しい Frame が作成されると、既存の Frame と置き換わり、この古い Frame はウィジェットによって維持されているリストで管理されます。アクティブな Frame が破棄されると、このリソースは NULL にセットされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

ウィジェットの背景のイメージです。このリソースが NULL の場合は背景色が使用されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

ウィジェットの背景のイメージの配置を指定します。有効な値は次の通りです。

XRTGA_IMAGE_LAYOUT_CENTERED: ウィジェットの中央に配置。イメージがウィジェットより大きい場合は、はみ出ている部分が切り取られます。

XRTGA_IMAGE_LAYOUT_TILED: イメージのコピーをタイルのように並べます。

XmNxrtGaInteractionCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge にインタラクションが起きたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。コールバックは *ActionStart()*、*ActionMove()*、*ActionEnd()*、*ResetCounter()* で呼び出され、XrtGaInteractionCallbackStruct のポインタが渡されます。

XmNxrtGaMapCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットに Map()アクションが登録されたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。XrtGaMapCallbackStruct のポインタが各 map コールバックに渡されます。

XmNxrtGaRenderList **XrtGaList** ***dynamic*** **G**

これと対のリソース、XmNxrtGaUseRederList が True の時は、ウィジェットはその子オブジェクトを XmNxrtGaRenderList に指定されている順序で描画します。オブジェクトが作成されると、これらは自動的に render リストに追加されます。従って、デフォルトでは、render リストは、子オブジェクトを作成された順に保持しています。このリソースは read-only ですが、このリストのポインタを取得し、直接リストを操作することができます。

XmNxrtGaRepaint **Boolean** **True** **CSG**

ウィジェットのリソースが変更された時に再描画するかどうか指定します。

XmNxrtGaResizeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットがリサイズされたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。XrtGaCallbackStruct のポインタが渡されます。

XmNxrtGaTextList **XrtGaList** **NULL** **G**

XrtGaText オブジェクトのリスト。新しく作成されたテキスト・オブジェクトは自動的に親のテキスト・リストに追加されます。このリストは read-only で内容を変更することはできません。

XmNxrtGaTickList **XrtGaList** **NULL** **G**

XrtGaTick オブジェクトのリスト。新しく作成された Tick オブジェクトは自動的に親の Tick リストに追加されます。このリストは read-only で内容を変更することはできません。

XmNxrtGaUseRenderList **Boolean** **False** **G**

このリソースの値が True の時は、ウィジェットはその子オブジェクトを XmNxrtGaRenderList のリストに指定されている順序で描画します。このリソースが False の時は、ウィジェットは子オブジェクトをデフォルトの順に描画します。ほとんどの場合は、デフォルトのままでオブジェクトは適した順序で描画されます。

3.5.2 Linear Gauge – その他のリソース

XmNxrtGaIndicatorList **XrtGaList** ***dynamic*** **G**

XmNxrtGaIndicator オブジェクトのリスト。新しく作成された Indicator オブジェクトは自動的に親の Indicator・リストに追加されます。このリストは read-only でその内容を変更することはできません。

XmNxrtGaLinearScaleList **XrtGaList** ***dynamic*** **G**

XrtGaLinearScale オブジェクトのリスト。新しく作成された Scale オブジェクトは、自動的に親の Linear Scale リストに追加されます。このリストは read-only でその内容を変更することはできません。

XmNxrtGaRangeList **XrtGaList** **NULL** **G**

XrtGaRange オブジェクトのリスト。(このオブジェクトがセットされている Scale は、XmNxrtGaLinearScaleList 上の Linear Scale でなければなりません。)新しく作成された Range オブジェクトは自動的の親の Range リストに追加されます。このリストは read-only でその内容を変更することはできません。

3.5.3 Linear Dial – その他のリソース

XmNxrtGaIndicator **XrtGaObject** ***dynamic*** **CG**

この Indicator の値は、オンとオフの(色またはイメージで表された)範囲の境界を示します。Indicator は、デフォルトで親ウィジェットに作成されます。新しく Indicator が作成されると既存の Indicator と置き換わり、古い Indicator はリスト上で維持されます。ただし、この古い Indicator がデフォルトである場合は破棄されます。現在アクティブな Indicator が破棄されると、このリソースは NULL にセットされます。

XmNxrtGaLinearScale **XrtGaObject** **NULL** **CG**

ウィジェットの Linear Scale。新しく Linear Scale が作成されると既存の Scale と置き換わり、古い Scale はリスト上で維持されます。現在アクティブな Linear Scale が破棄されると、このリソースは NULL にセットされ、ウィジェットには何も描画されません。

XmNxrtGaOffColor **Pixel** ***inherited**** **CSG**

Linear Dial のオフの色。

*XmNxrtGaOffColor のデフォルトの色は、ウィジェットの背景色から継承されます。

XmNxrtGaOffImage **XrtGaImage** **NULL** **CSG**

Linear Dial のオフの部分の背景のイメージ。この値が NULL の時はオフの色が使用されます。

XmNxrtGaOnColor **Pixel** ***inherited*** **CSG**

Linear Dial のオンの色。

*XmNxrtGaOnColor のデフォルトの色は、ウィジェットの前景色から継承されます。

XmNxrtGaOnImage **XrtGaImage** **NULL** **CSG**

Linear Dial のオンの部分の背景のイメージ。この値が NULL の時はオンの色が使用されます。

XmNxrtGaShape **XrtGaShape** **XRTGA_SHAPE_RECTANGLE** **CSG**

オンとオフの範囲の形。有効な値は、XRTGA_SHAPE_RECTANGLE、XRTGA_SHAPE_CIRCLE、XRTGA_SHAPE_THERMOMETER、XRTGA_SHAPE_TANK です。XRTGA_SHAPE_IMAGE はこのリソースでは使用できません。

3.6 手続きとマクロ

Linear Dial と Linear Gauge の手続きとマクロのリストです。

XmCreateXrtGaLinearDial()

Linear Dial を作成します。

```
Widget
XmCreateXrtGaLinearDial (
      Widget  parent,
      char *  name,
      Arg *   arglist,
      Cardinal argcount
      )
```

parent は作成される XrtGaLinearDial の親、*name* は作成されたウィジェットの名前です。*arglist* は、作成時に XrtGaLinearDial に渡される XrtArgVal リソースと値のペアのリストです。このリストは NULL とすることもできます。*argcount* は *arglist* のエレメント数です。

XmIsXrtGaLinearDial()

```
XmIsXrtGaLinearDial(w)
```

w が XrtGaLinearDial ウィジェットまたは XrtGaLinearDial ウィジェットのサブクラスであるときに True を返すマクロです。

XmCreateXrtGaLinearGauge()

Linear Gauge ウィジェットを作成します。

```
Widget
XmCreateXrtGaLinearGauge (
      Widget  parent,
      char *  name,
      Arg *   arglist,
      Cardinal argcount
      )
```

parent は作成される XrtGaLinearGauge の親、*name* は作成されたウィジェットの名前です。*arglist* は、作成時に XrtGaLinearGauge に渡される XrtArgVal リソースと値のペアのリストです。このリストは NULL とすることもできます。*argcount* は *arglist* のエレメント数です。

XmIsXrtGaLinearGauge()

```
      XmIsXrtGaLinearGauge(w)
```

w が XrtGaLinearGauge ウィジェットまたは XrtGaLinearGauge ウィジェットのサブクラスであるときに True を返すマクロです。

4

LEDIcon ウィジェット

はじめに ■ LEDIcon のリソース
トランスレーションとアクション ■ サンプル・コード
ウィジェットの概要 ■ リソースの一覧
手続きと方法

4.1 はじめに

LEDIcon ウィジェットは、ソフトウェアのプログレス・メータやオーディオのスペクトラム・アナライザに似ています。ウィジェットは Scale を持ち、値は Icon と呼ばれる円形または矩形のマークによって示されます。これらのマークの *オン* と *オフ* の状態は、色やイメージによって表されます。ウィジェットの値が増加するにつれ、*オン* の色を表示する Icon が増えていきます。値がかなり速い速度で変動し、最大値を表示しておいた方が便利な場合は、一番高い値の *オン* の Icon をしばらく表示したままにしておくことができます。

LEDIcon は垂直、水平どちらの向きにセットすることもでき、Scale 値が増加する方向も同様にセットできます。

(Icon の数によって数値を表す)マグニチュード・モードでは、Icon がオンになる値は *LEDStartValue* と呼ばれます。この値は 0 でなくても構いませんが、必ず整数にしてください。ウィジェットの値パラメータが LEDStartValue を超えると、値が 1 増える毎に別の Icon がオンとなっていきます。バイナリ・モードでは、Icon は二進法で求められた数値を表示します。正の数は、方向が *forward*、向きが *horizontal* であれば、左から右に向かって Icon のオンとオフで表されます。負の数は、2 の補数を使用して表されます。例えば、バイナリ・モードでは、-1 は各 Icon を全てオンにすることによって表されます。

4.2 LEDIcon のリソース

継承されたリソース

継承されているリソースは、XmNbackground、XmNforeground、XmNxrtGaBorder です。これらのリソースは、この章の最後のリソースの一覧に解説があります。

共通なリソース

共通なリソースは、XmNxrtGaDebug、XmNxrtGaDoubleBuffer、XmNxrtGaRepaint です。これらのリソースは、この章の最後のリソースの一覧に解説があります。

値の設定

LEDIcon ウィジェットの現在値は、XmNxrtGaLEDValue リソースを使用してセットします。ウィジェットに値を正しく表示させるには、値は必ず整数にし、XmNxrtGaLEDStartValue と下記の数式で得られる数値との間の数にしてください。

NumberOfIcons - LedStartValue

この式の *NumberOfIcons* は Icon の数、*LedStartValue* は、XmNxrtGaLEDStartValue に現在セットされている値を指します。

LEDIcon ウィジェットの Scale には、最大値をセットする必要はありませんが、XmNxrtGaLEDValue に与えられる数に対して十分な数の Icon を用意してください。例えば、-5 から 20 の値を表示する場合は 26 個の Icon が必要です。0 から 100 の値を表示しなければならないのに、アプリケーション側の GUI の制限により 10 個の Icon しか使用できない場合は、10 個の Icon で表現できるように値を割り当てる必要があります。

最大値

変動する値が最大値に達したときに、それを一次的にでも表示しておいた方がよい場合があります。その場合は、XmNxrtGaMaxIcon を True にセットしてください。値が動的に変化している状況では、ウィジェットによって表示されている値が減少すると、一番最後に達した最大値に対応する Icon が XmNxrtGaDecayRate で指定されているミリ秒間、表示されたままになります。その後、その次の Icon が順々にオンになり、現在値に達します。最大値に到達後直ちに値が下がっていく場合、この最大値を表示しておく方法は、最大値から、XmNxrtGaDecayRate によって指定された時間毎にその中間の Icon を通過して現在値に達する過程を視覚的に表します。

バイナリ・モード

Icon のカウントによって値を表示する他、XRT/gauge では、XmNxrtGaDisplayMode を使用してバイナリ・モードで表示することができます。このリソースが XRTGA_DISPLAY_MODE_BINARY にセットされると、表示される数はバイナリで表現されます。例えば、XmNxrtGaLEDValue が 5 で、*horizontal* と *forward* の方向が指定されている場合、左から三つの Icon はそれぞれ、オン、オフ、オンとなります。

それ以上の数の Icon がある場合は、それらは全てオフになります。また、方向を *backward* にすると、数値の表示が逆になります。

負の数は 2 の補数を使用して表されます。最初に、その負の数に対応する正の数のバイナリを求め、次にその Icon のオンとオフを全て反転します。その後、その反転された値に 1 をバイナリ加算します。

XmNxrtGaLEDStartValue は、バイナリ・モードでは使用されません。

LEDIcon のサイズとレイアウト

Icon が並ぶ向きを設定して、Icon が行または列の形に配置されるようにできます。XmNxrtGaOrientation は XmHORIZONTAL または XmVERTICAL に設定できます。また、XmNxrtGaDirection を使用すれば、値が増加する方向を指定することができます。向きが horizontal で、XmNxrtGaDirection が XRTGA_DIRECTION_FORWARD であれば、値は右方向に増加していきます。向きが vertical で XmNxrtGaDirection が XRTGA_DIRECTION_BACKWARD であれば、値は下方向に増加していきます。

Icon 間のスペースは XmNxrtGaIconSpacing でセットします。このリソースは、Icon の間隔を pixel で指定します。この設定は、horizontal、vertical の両方に別々に設定できます。

ウィジェットのサイズに応じて Icon 自身の大きさを調整する方法は XmNxrtGaSizingPolicy でコントロールします。デフォルトでは、このリソースの値は XRTGA_SIZING_POLICY_WIDGET で、Icon はウィジェットの縦横方向に一杯になるように表示されます。

イメージがオンとオフの表記に使用され、上記のリソースが XRTGA_SIZING_POLICY_ICON であると、Icon は、使用されている一番大きなイメージに基づいて Icon の理想的なサイズを計算し、このサイズをウィジェットに渡します。もし、ウィジェットが小さい場合は、親にウィジェットのサイズを大きくするように要求します。この要求が受け入れられると、ウィジェットがリサイズされます。ウィジェットが、Icon の理想的なサイズに対して十分大きい場合は、ウィジェットのサイズは調整されません。

このリソースが XRTGA_SIZING_POLICY_BY_POSITION の場合は、ウィジェット内の Icon の位置は、下記のリソースによって決定されます。

XmNxrtGaXOrigin	Icon のグループは、このリソースによって指定された pixel 数分だけ横方向にインデントされる。
XmNxrtGaYOrigin	Icon のグループは、このリソースによって指定された pixel 数分だけ縦方向にインデントされる。
XmNxrtGaIconWidth	全ての Icon に共通の pixel 幅。幅の合計がウィジェットより大きい場合は、Icon は右から左方向へ切り取られる。
XmNxrtGaIconHeight	全ての Icon の共通の pixel の高さ。高さの合計がウィジェットより大きい場合は、Icon は下から上方向へ切り取られる。

このリソースの値が、XRTGA_SIZING_POLICY_CENTER であれば、ウィジェット内の Icon の位置は、XmNxrtGaIconHeight と XmNxrtGaIconWidth によって決定され、Icon はウィジェット内で中央に配置されます。

4.3 トランスレーションとアクション

LEDIcon のトランスレーションは、次の通りです。

```
<Btn1Down>:   ActionStart(Icon)
<Btn1Motion>: ActionMove(Icon)
<Btn1Up>:     ActionEnd(Icon)
```

4.4 サンプル･コード

ウィジェットは、Icon のリストを XmNxrtGaIconList で管理します。次のコードは、このリストを取得して現在作成されている Icon の数を調べ、最後のアイテムを除外しています。

```
Widget    led;     /* The LEDIcon widget */
Widget    icon;    /* Icon object variable
int count;         /* Variable that holds
                   the number of items in the list */
XrtPDSList list;

XtVaGetValues(led,
      XmNxrtGaIconList, &list,
      NULL);
count = XrtPDSListGetItemCount(list);
icon = *(XrtGaObject *)XrtPDSListGetItem(list, count - 1);
XtDestroyWidget((Widget) icon);
```

Icon を破棄すると、この Icon は自動的にリストから削除されます。各 Icon にはイメージを指定することができる他、ウィジェットの背景にもイメージを指定することができます。この場合、XmNxrtGaImage を使用し、イメージの配置は XmNxrtGaImageLayout でセットしてください。このリソースのデフォルト値は XRTGA_IMAGE_LAYOUT_CENTERED ですが、XRTGA_IMAGE_LAYOUT_TILED に設定することもできます。タイル型を選択した場合は、イメージが複数コピーされ、それらがウィジェット全体を占めるように配置されます。_CENTERED が設定され、イメージがウィジェットより大きい場合、イメージはリサイズされずにはみ出た部分は切り取られます。

コールバック

このウィジェットに定義されているコールバックは、XmNxrtGaExposeCallback、XmNxrtGaInteractionCallback、XmNxrtGaResizeCallback、XmNxrtGaValueChangedCallback です。詳細は第 22 章を参照してください。

4.5 ウィジェットの概要

LEDIcon widget Include File *<Xm/XrtGaLEDIcon.h>*

Class Name: XmXrtGaLEDIcon

Class Hierarchy: Core ➔ XmPrimitive ➔ XmXrtGaLEDIcon

Class Pointer: xmXrtGaLEDIconWidgetClass

4.6 リソースの一覧

LEDIcon の全てのリソース及び、よく使用される継承されたリソースの一覧です。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。

XmNbackground **Pixe** ***Inherited*** **CSG**

ウィジェットの背景色です。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_3D_OUT** **CSG**

ウィジェットの枠線の種類です。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースからの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDecayRate **int** **500** **CSG**

`XmNxrtGaMaxIcon` が `True` でマグニチュード・モードの場合、このリソースはオンになった最大の Icon が消える速度をミリ秒で指定します。

XmNxrtGaDisplayMode **XrtGaDisplayMode** **XRTGA_DISPLAY_MODE_MAGNITUDE** **CSG**

Icon は、マグニチュードとバイナリの二つのモードで表示することができます。`XRTGA_DISPLAY_MODE_MAGNITUDE` の場合は、`XmNxrtGaLEDValue` の値がオンになる Icon の数を決定します。`XRTGA_DISPLAY_MODE_BINARY` の場合は、bit で解釈された値がどの Icon をオンまたはオフにするか決定します。例えば、値 9 はバイナリ表記では 1001 となり、この場合、0 番目と 3 番目の Icon がオンになります。

XmNxrtGaDirection **XrtGaDirection** **XRTGA_DIRECTION_FORWARD** **CSG**

min から *max* への方向を指定します。有効な値は、`XRTGA_DIRECTION_FORWARD` と `XRTGA_DIRECTION_BACKWARD` です。LEDIcon では、`XmNxrtGaOrientaiton` でコントロールされる向きによって、forward は左から右または下から上を意味します。

XmNxrtGaDoubleBuffer **Boolean** **True** **CSG**

ウィジェットの描画をダブル・バッファリングするかどうか指定します。ダブル・バッファリングは、オブジェクトが描画または再描画されるときのちらつきを減少させるテクニックです。

XmNxrtGaExposeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge が再描画を行ったことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。XrtGaCallbackStruct のポインタが渡されます。

XmNxrtGaIconHeight **Dimension** **10** **CSG**

Icon・リストの各 Icon の高さです。

XmNxrtGaIconList **XrtGaList** **NULL** **G**

XrtGaIcon オブジェクトの XrtGaList に指定されている Icon・リストです。このリソースが NULL の場合は何も描画されません。新しい Icon・オブジェクトが作成されると、親の Icon・リストに自動的に追加されます。このリストは read-only で内容を操作することはできません。

XmNxrtGaIconSpacing **Dimension** **2** **CSG**

Icon の間隔を pixel で指定します。

XmNxrtGaIconWidth **Dimension** **10** **CSG**

Icon・リストの Icon の幅です。デフォルト値は、サイズの決定法に依存します。サイズの決定法が XRTGA_SIZING_POLICY_WIDGET であれば、値はウィジェットのサイズ、Icon の数、Icon の間隔から計算されます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

ウィジェットの背景のイメージです。この値が NULL の場合は、背景色が使用されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージの配置を指定します。有効な値は、XRTGTA_IMAGE_LAYOUT_CENTERED と XRTGA_IMAGE_LAYOUT_TILED です。

XmNxrtGaInteractionCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge にインタラクションが起きたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。コールバックは、*ActionStart()*、*ActionMove()*、*ActionEnd()*、*ResetCounter()*で呼び出されます。XrtGaInteractionCallbackStruct のポインタが渡されます。

XmNxrtGaInteractionType **XrtGaInteractionType** **XRTGA_INTERACTION_TYPE_NONE** **CSG**

このウィジェットでは、XmNxrtGaInteractionType は Boolean のように機能します。インタラクションは、XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE または XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE をセットすることによりアクティブになります。一度、アクティブになると、動作は XmNxrtGaDisplayMode に依存します。

表示モードの設定	XmNxrtGaInteractionType の動作
XRTGA_DISPLAY_MODE_MAGNITUDE	Icon をクリックすると、その Icon とそれより小さい値の Icon が全てオンになる。ドラッグすると、マウスがある Icon とそれより小さい値の Icon が全てオンになる。

表示モードの設定	XmNxrtGaInteractionType の動作
XRTGA_DISPLAY_MODE_BINARY	マウス・クリックによって Icon がオンまたはオフになる。ドラッグ・アクションが起きた Icon は、全てオンとオフがトグルされる。

XmNxrtGaLEDStartValue **int** **0** **CSG**

LEDIcon の最小値を指定します。現在の XmNxrtGaLEDValue が XmNxrtGaLEDStartValue と同じかそれより小さいと、全ての Icon・オブジェクトは XmNxrtGaOffColor または XmNxrtGaOffImage を使用して表示されます。

XmNxrtGaLEDValue **int** **0** **CSG**

LEDIcon をオンまたはオフにする、ウィジェットの現在値です。オンの状態の Icon の数は、このリソースの値から *start* 値を引いた数です。従って、ウィジェットの現在値が 5 で、XmNxrtGaLEDStartValue の値が 3 であれば、二つの Icon がオンになり、他は全てオフになります。

XmNxrtGaMaxIcon **Boolean** **False** **CSG**

このリソースが True でマグニチュード・モードの場合、オンになっている最大の Icon は、XmNxrtGaDecayRate で指定されたミリ秒表示されたままになります。

XmNxrtGaOrientation **unsigned char** **XmHORIZONTAL** **CSG**

ウィジェットの向きを指定します。有効な値は XmHORIZONTAL と XmVERTICAL です。horizontal の場合はプロセスの進行具合を表示するプログレス・バー、vertical の場合は、スペクトラム・アナライザのように表示されます。

XmNxrtGaRepaint **Boolean** **True** **CSG**

ウィジェットのリソースが変更された時、ウィジェットを再描画するかどうか指定します。

XmNxrtGaResizeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge がリサイズされたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。XrtGaCallbackStruct のポインタが渡されます。

XmNxrtGaSizingPolicy XrtGaSizingPolicy XRTGA_SIZING_POLICY_WIDGET CSG

Icon のサイズに基づいてウィジェットのサイズをコントロールします。有効な値は次の通りです。

XRTGA_SIZING_POLICY_WIDGET	ウィジェットの幅と高さが Icon のサイズを決定し、ユーザは Icon をリサイズすることはできない。
XRTGA_SIZING_POLICY_ICON	Icon の幅と高さがウィジェットのサイズを決定し、ユーザは Icon をリサイズすることはできない。
XRTGA_SIZING_POLICY_BY_POSITION	XmNxrtGaIconWidth と XmNxrtGaIconHeight が Icon のサイズを決定する。ウィジェットが Icon より小さい場合、はみ出た部分は切り取られる。
XRTGA_SIZING_POLICY_CENTER	XmNxrtGaIconWidth と XmNxrtGaIconHeight が Icon のサイズを決定する。Icon のグループは、ウィジェット内で中央に配置される。ウィジェットが Icon より小さい場合、はみ出た部分は切り取られる。

XmNxrtGaSpacingHorizontal int 10 CSG

Icon・グループとウィジェットの左と右側の枠とのスペースを pixel で指定します。

XmNxrtGaSpacingVertical int 5 CSG

Icon・グループの上部と下部のスペースを pixel で指定します。このリソースは、orientation(向き)とは関係なく常に縦のスペースを指します。

XmNxrtGaValueChangedCallback XtCallbackList NULL CSG

値が変更された時に呼び出されるコールバックのリストです。XmNxrtGaValueChangedCallbackStruct のポインタが渡されます。

XmNxrtGaXOrigin Position 0 CSG

サイズの決定法が XRTGA_SIZING_POLICY_BY_POSITION の場合、この値は一番目の Icon の x 座標を指します。

XmNxrtGaYOrigin Position 0 CSG

サイズの決定法が XRTGA_SIZING_POLICY_BY_POSITION の場合、この値は一番目の Icon の y 座標を指します。

4.7 手続きとマクロ

XmCreateXrtGaLEDIcon()

LEDIcon ウィジェットを作成します。

```
Widget
XmCreateXrtGaLEDIcon(
 Widget       parent,
 char *       name,
 Arg          *arglist,
 Cardinal     argcount
 )
```

parent は親、*name* は新しく作成されるウィジェットの名前です。*arglist* は作成時に XrtGaLEDIcon に与えられる XrtArgVal リソースと値のペアのリストで、NULL とすることもできます。この手続きは、新しく作成された XrtGaLEDIcon を返しますが、ウィジェットが作成されなかった場合は NULL を返します。

XmIsXrtGaLEDIcon()

```
XmIsXrtGaLEDIcon(w)
```

w が XrtGaLEDIcon ウィジェットまたは XrtGaLEDIcon のサブクラスである場合は、True を返すマクロです。

5

LEDNumerical ウィジェット

はじめに ■ LEDNumerical ウィジェットの値
ウィジェットと表示セグメントの色 ■ サイズ
クロック表示 ■ 数値の傾き
コールバック ■ サンプル・コード
ウィジェットの概要 ■ リソースの一覧
手続きと方法

5.1 はじめに

LEDNumerical ウィジェットは、七つのセグメントを使用して数値または文字数字(alphanumeric)を表します。

このウィジェットは、スコアボードや、時間のデジタル表示等、単純に数値を表示する場合に適しています。

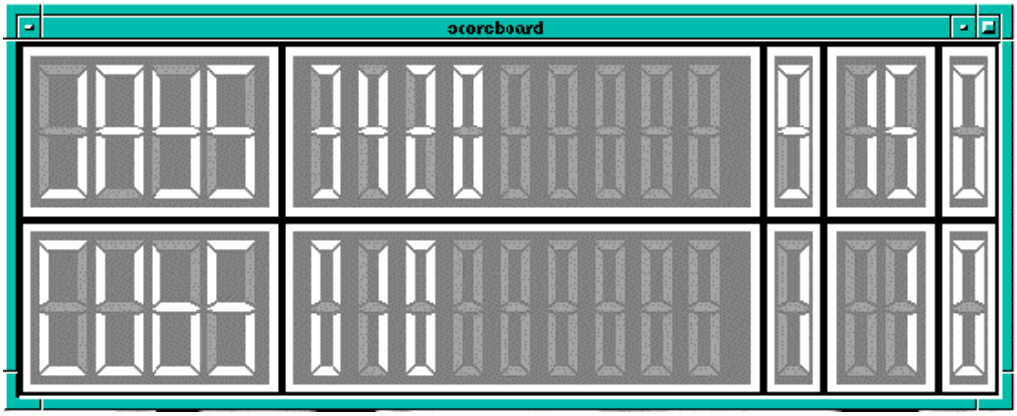


図10　LEDNumerical ウィジェットで作成された野球のスコアボード

5.2 LEDNumerical ウィジェットの値

LEDNumerical ウィジェットは数値または文字数字(alphanumeric)の値を表示します。XmNxrtGaValueType を使用して、XRTGA_VALUE_TYPE_NUMERIC と XRTGA_VALUE_TYPE_ALPHANUMERIC のどちらかを選択できます。XRTGA_VALUE_TYPE_ALPHANUMERIC を選択すると、7 つのセグメントから構成される LED digit という点を考慮しながら、XmNxrtGaText に与えられたストリングを表示します。

XRTGA_VALUE_TYPE_NUMERIC の場合は、XmNxrtGaLeadingPlusMinus を True にして、数値の前にプラスやマイナスの記号をつけたり、XmNxrtGaLeadingZeros を True にして、実際に表示する数値の前に(ブランクの代わりに)0 を表示することができます。

表示する桁数は、XmNxrtGaNumDigits でセットしてください。

5.2.1 小数点

XmNxrtGaFixedDecimal が True で、値の型が数値の場合、XmNxrtGaDecimalPlaces を使用して小数点以下の桁数を指定することができます。

5.3 ウィジェットと表示セグメントの色

値を表示する部分のセグメントには、オンとオフの色を XmNxrtGaOnColor と XmNxrtGaOffColor で指定することができます。これらのリソースがセットされていない場合は、親ウィジェットの前景色がオン、背景色がオフに使用されます。ウィジェットの背景色は、LED の 7 つのセグメントのセル以外の部分に使用されます。

上記の方法でオフの色を設定する以外に、XmNxrtGaColorMixPercentage を使用して、ウィジェットの背景色とセグメントのオンの色を混ぜ合わせたオフの色をセットすることができます。このリソースには、背景色と混ぜ合わせるオンの色の割合を整数で指定します。例えば、この値が 20 であると、オンの色が 20%で背景色が 80%の割合で作成された色がオフの色として使用されます。

5.4 サイズ

LEDNumerical ウィジェットは、与えられた幅と高さのパラメータに基づいてサイズを決定します。XmNheight はウィジェットの高さを、XmNwidth が幅を指定します。ここで指定する幅は、XmNxrtGaNumDigits で指定されている桁を表すのに十分な大きさにしてください。ウィジェットのサイズが小さすぎると、表示が途中で切れてしまいます。表示部分のセグメントの幅は XmNxrtGaSegmentWidth で指定され、その他の幅の設定には影響されません。

図 11 は、一桁の数値を表すウィジェットです。ハイライトの幅、枠線の幅、横方向のスペースが実際に表示する桁の幅に影響します。

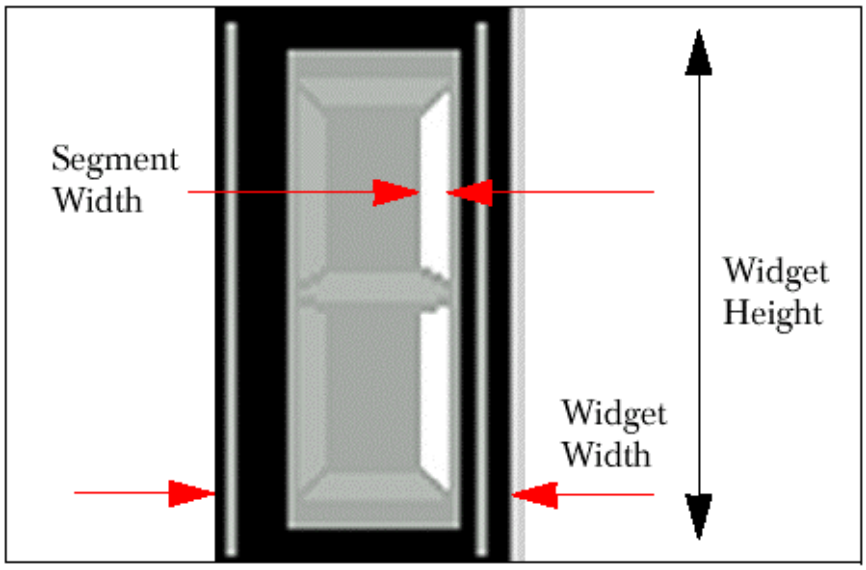


図11　一桁の数値を表示する LEDNumerical ウィジェット

図 12 は、ウィジェットの幅を大きくした場合です。可変のパラメータは数値の幅だけですので、ウィジェットの幅に応じて大きくなります。

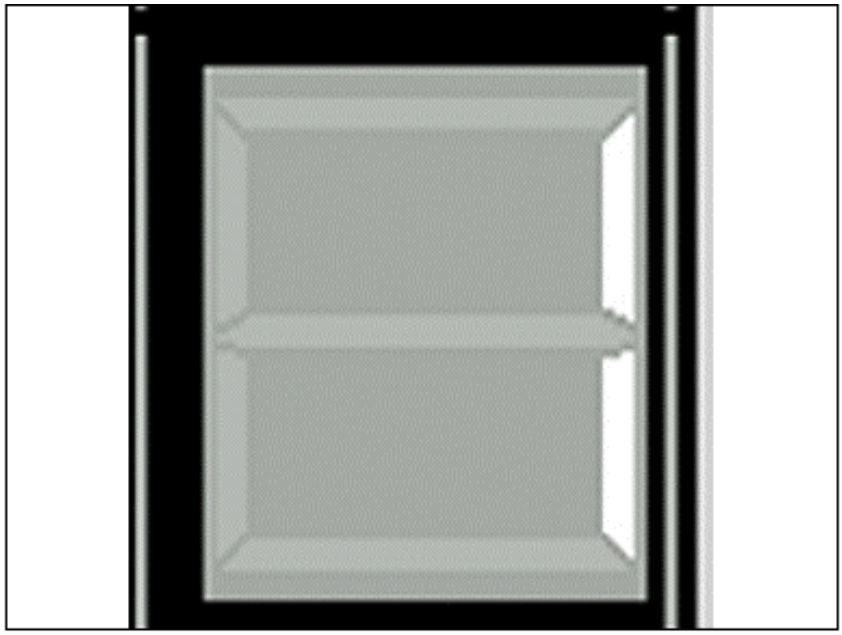


図12　他のサイズはそのままでウィジェットの幅を大きくした場合

ウィジェットと数値との間は `XmNxrtGaSpacingHorizontal` でコントロールします。図 13 で"JayS"と表示しているウィジェットには、このリソースに大きな値を設定し、その下の"CubS"と表示しているウィジェットには、小さな値を設定しています。

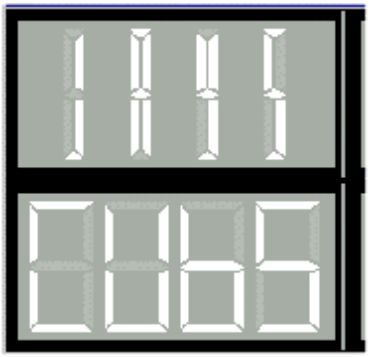


図13　異なるスペースを設定した場合の二つのウィジェット

サイズを決定するリソースは、pixel で表される長さをパラメータとしてとります。

5.5　クロック表示

(12 時間表示の)時刻を表す 10 の桁には、0 か 1 しか必要ありません。一般の多くの LED 表示では、この桁には(七つではなく)二つのセグメント表示が使用され、1 はオン、0 はオフというように表示されています。XRT/gauge の LEDNumerical で同様な表示を行うには、`XmNxrtGaLeadingOne` を `True` にし、LED の最初の桁が二つのセグメント型になるようにしてください。

5.6　数値の傾き

数値はデフォルトでは真っ直ぐに表示されていますが、`XmNxrtGaSlopeOffset` をセットして右または左に傾けることができます。

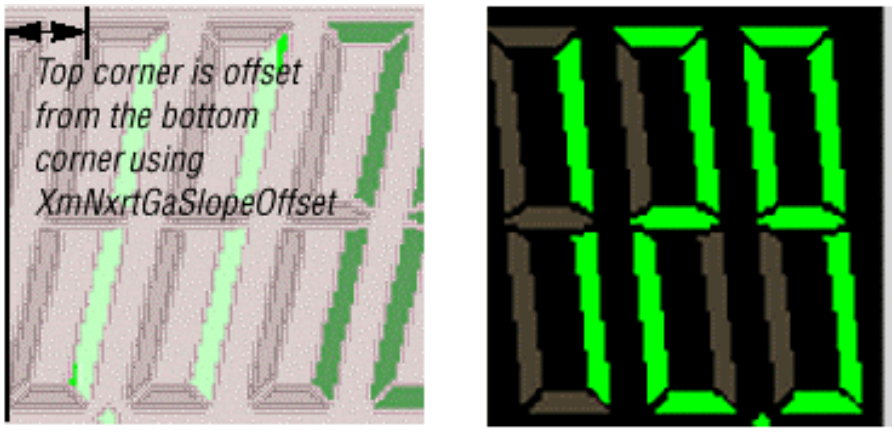


図14　数値を傾けて表示したウィジェット

図 14 では、数値が斜体で表示されています。左側のウィジェットではオフセットのパラメータに 20pixel が指定されています。右側の黒い背景のウィジェットでは、オフセットに-10 が設定されています。

5.7 コールバック

ウィジェットには、XmNxrtGaExposeCallback、XmNxrtGaResizeCallback、XmNxrtGaValueChangedCallback の三つのコールバックが定義されています。詳細は第 22 章を参照してください。

5.8 サンプル･コード

アプリケーションで LEDNumerical ウィジェットを作成するには、リソースを次のようにセットしてください。

```
/* Create LED Numerical Widget*/
value_type = XRTGA_VALUE_TYPE_ALPHANUMERIC;
border = XRTGA_BORDER_PLAIN;
 jays =
     XtVaCreateManagedWidget("jays",
             xmXrtGaLEDNumericalWidgetClass,form,
             XmNheight, 100,
             XmNwidth, 150,
             XmNxrtGaValueType,           value_type,
             XmNxrtGaText,                "JAYS",
             XmNxrtGaSegmentWidth,        5,
             XmNxrtGaSpacingHorizontal,   5,
             XmNxrtGaSpacingVertical,     5,
             XmNxrtGaNumDigits,           4,
             XmNxrtGaBorder,              border,
             XmNshadowThickness,          5,
             XmNxrtGaColorMixPercentage,  20,
             COLOR_RES(XmNborderColor,    "black"),
             COLOR_RES(XmNxrtGaOnColor,   "white"),
             XmNtopAttachment,            XmATTACH_FORM,
             XmNleftAttachment,           XmATTACH_FORM,
             COLOR_RES(XmNbackground,     "darkgreen"),
             NULL);
```

上記のコードにより、図 10 の左上に表示されているウィジェットと同様なウィジェットが作成されます。

5.9 ウィジェットの概要

Include File:	*<Xm/XrtGaLEDNumerical.h>*
Class Name:	XmXrtGaLEDNumerical
Class Hierarchy:	Core ➔ XmPrimitive ➔ XmXrtGaLEDNumerical
Class Pointer:	xmXrtGaLEDNumericalWidgetClass

Motif ウィジェットの階層や、LEDNumerical ウィジェットの階層内での位置に関しては、図 1 を参照してください。

5.10 リソースの一覧

LEDNumerical ウィジェットの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

ウィジェットの背景色。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_3D_OUT** **CSG**

ウィジェットの枠線の種類。枠線の幅は `XmNshadowThickness` で指定します。

XmNxrtGaColorMixPercentage **int** **30** **CSG**

`XmNbackground` と `XmNxrtGaOnColor` を混ぜ合わせてオフの色を指定します。例えば、値が 30 であれば、オンの色 30%と背景色 70%を混合した色がオフの色として指定されます。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースからの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDecimalPlaces **int** **0** **CSG**

`XmNxrtGaFixedDecimal` が `True` で値の型が数値の場合、このリソースが小数点以下の桁数を指定します。

XmNxrtGaDecimalPosition **Position** **0** **CSG**

テキストのベースラインに対する小数点の位置または、中央からのコロンの位置を pixel で指定します。

XmNxrtGaDecimalSize **Dimension** **5** **CSG**

小数点やコロンのサイズを pixel で指定します。

XmNxrtGaDoubleBuffer **Boolean** **True** **CSG**

ウィジェットの描画をダブル・バッファリングするかどうか指定します。ダブル・バッファリングは、オブジェクトが描画または再描画される時に起こるちらつきを減少させるテクニックです。

XmNxrtGaExposeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットが再描画したことをアプリケーションに伝えるときに呼び出されるコールバックのリストを指定します。`XrtGaCallbackStrucnt` のポインタが渡されます。

XmNxrtGaFixedDecimal **Boolean** **False** **CSG**

このリソースが `True` で値の型が数値の場合、`XmNxrtGaDecimalPlaces` を使用して小数点の位置を指定します。

XmNxrtGaLeadingOne **Boolean** **False** **CSG**

先頭の桁に数値の 1 を表示します。(値またはストリングが存在していればオン、存在していなければオフになります。)

XmNxrtGaLeadingPlusMinus **Boolean** **False** **CSG**

値の型が数値で 0 でない場合または、型が文字数値(alphanumeric)で'+'または'-'の記号があるときには、先頭にプラスまたはマイナスの記号が表示されます。

XmNxrtGaLeadingZeros **Boolean** **False** **CSG**

数値モードの時に先頭に 0 を表示します。`False` の場合、この場所はブランクです。

XmNxrtGaNumDigits **int** **5** **CSG**

表示される桁数を指定します。先頭のプラスやマイナスの記号は含まれません。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

ウィジェットの背景のイメージ。`NULL` の場合は、背景色が使用されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージの配置を指定します。有効な値は、`XRTGA_IMAGE_LAYOUT_CENTERED` と `XRTGA_IMAGE_LAYOUT_TILED` です。

XmNxrtGaOffColor **Pixel** ***dynamic**** **CSG**

オフの色です。この色がセットされていない場合は、親ウィジェットの背景色が使用されます。

** デフォルト値は `XmNxrtGaColorMixPercentage` 値に依存します。`XmNxrtGaColorMixPercentage` がデフォルトのままであれば、オフの色は on_color の 30%と背景色 70%を混合した色です。*

XmNxrtGaOnColor **Pixel** ***inherited*** **CSG**

オンの色です。この色がセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaRepaint **Boolean** **True** **CSG**

リソースが変更された時にウィジェットを再描画するかどうか指定します。

XmNxrtGaResizeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットがリサイズされたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。`XrtGaCallbackStruct` のポインタが渡されます。

XmNxrtGaSegmentSpacing **Dimension** **2** **CSG**

LED のセグメント間のスペースを pixel で指定します。

XmNxrtGaSegmentWidth **Dimension** **5** **CSG**

LED セグメントの幅を pixel で指定します。

XmNxrtGaSlopeOffset **int** **0** **CSG**

LED の表示の左上角が横方向に何 pixel 分、左下角からオフセットするか指定します。このリソースにより、LED が斜体に表示されます。

XmNxrtGaSpacingHorizontal **Dimension** **10** **CSG**

各桁の間のスペースを pixel で指定します。

XmNxrtGaSpacingVertical **Dimension** **2** **CSG**

LED の上部と下部のスペースを指定します。

XmNxrtGaText **String** **NULL** **CSG**

値の型が文字数字(alphanumeric)であれば、このリソースによって指されるテキスト・ストリングが LED 値の表示に使用されます。このリソースの値が NULL の場合、全ての桁は空白のままになります。

XmNxrtGaValue **double** **0.0** **CSG**

値の型が数値であれば、このリソースが LED 値の表示に使用されます。

XmNxrtGaValueChangedCallback **XtCallbackList** **NULL** **CSG**

値が変更された時に呼び出されるコールバックのリストです。XmNxrtGaValueChangedCallbackStrucnt のポインタが渡されます。

XmNxrtGaValueType **XrtGaValueType** **XRTGA_VALUE_TYPE_NUMERIC** **CSG**

使用される値の型、数値か文字数字(alphanumeric)かを指定します。数値(XRTGA_VALUE_TYPE_NUMERIC)であれば、値は XmNxrtGaValue から取得されます。文字数字(XRTGA_VALUE_TYPE_ALPHANUMERIC)であれば、値は XmNxrtGaText から取得されます。

5.11 手続きとマクロ

XmCreateXrtGaLEDNumerical()

LEDNumerical ウィジェットを作成します。

```
Widget
XmCreateXrtGaLEDNumerical(
      Widget       parent,
      char *       name,
      Arg          *arglist,
      Cardinal     argcount
      )
```

parent は親、*name* は新しく作成されるウィジェットの名前です *arglist* は作成時に XrtGaLEDNumerical に提供される XrtArgVal のリソースと値のペアで、これは NULL とすることもできます。*argcount* は引数の数です。この手続きは、新しく作成された XrtGaLEDNumerical ウィジェットを返しますが、ウィジェットが作成されなければ NULL を返します。

XmIsXrtGaLEDNumerical()

```
XmIsXrtGaLEDNumerical(w)
```

w が XrtGaLEDNumerical ウィジェットまたは XrtGaLEDNumerical ウィジェットのサブクラスであれば True を返すマクロです。

6

Percent ウィジェット

Percent ウィジェット ■ スケールの選択
ラベル ■ Item 上にラベルを配置
Percent ウィジェットのリソースについての注意
トランスレーションとコールバック■ ウィジェットの概要
リソースの一覧■ 手続きと方法

6.1 Percent ウィジェット

Percent ウィジェットは、ある関連する Item のグループがそれぞれ全体の何パーセントを占めるか表示するウィジェットです。

サンプル・コード

下記のサンプル・コードは Percent ウィジェットを作成し、幾つかのリソースをセットしています。

```
Widget scale; /* The Percent widget's associated scale */
 percent =
     XtVaCreateManagedWidget("percent",
         xmXrtGaPercentWidgetClass,   form,
         XmNxrtGaScale,               scale,
         XmNxrtGaCalcMode,            XRTGA_CALC_MODE_PERCENT,
         NULL);
```

Circular Scale を持つ Percent ウィジェットは円グラフのように表示されます。円グラフは、全体に対する割合を示す場合に最もよく使用される形です。これが Linear(線形)で表されると、各棒の幅が割合を表す棒グラフのようになります。

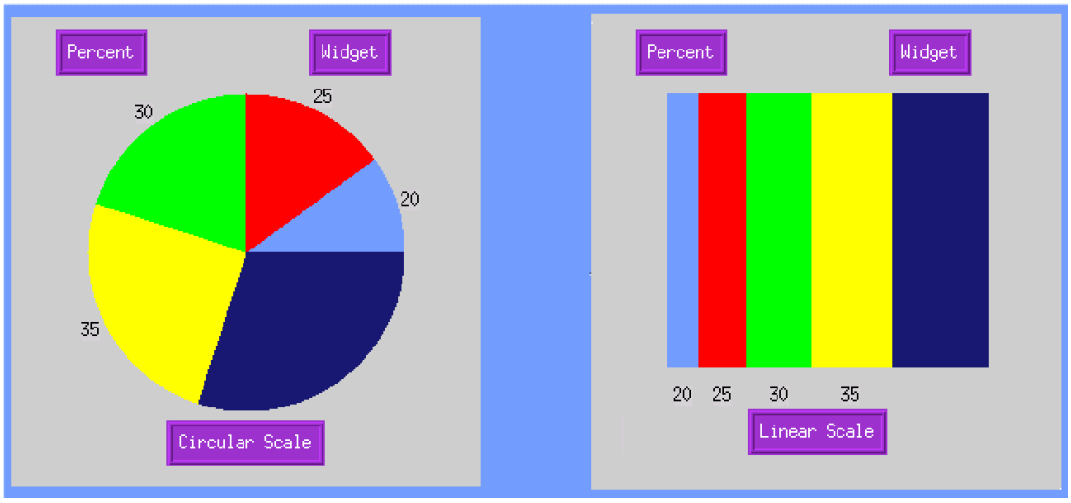
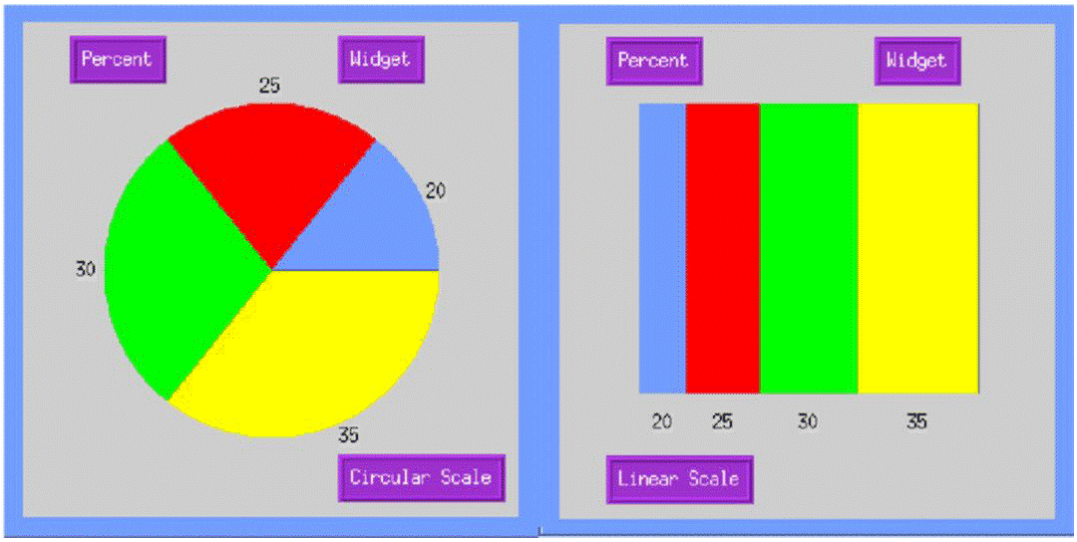


図15　Circular と Linear Scale で表示された Percent ウィジェット

図 15 は、*calculation-by-percent* モードの時に作成できるウィジェットです。両方のウィジェットで色が暗い部分は'left over'と呼ばれる部分で、Item 値の合計が Scale で表す範囲より小さい場合に作成されます。leftover の部分は、ウィジェットの `XmNxrtGaItemList` の一部としてアクセスできる内部 Item に記憶されます。この値は、Scale の *max - min* 値と Item の値の合計の差です。

もう一つの方法として、leftover が表示されないように計算する方法があります。*calculation-by-ratio* モードの時は、leftover の部分は無視して Item だけを計算の対象とし、各 Item が Item の合計の何割を締めるかを計算します。図 16 は、この計算法で示した Scale です。



.図16　raio モードの Percent ウィジェット

Percent ウィジェットで、Item の値がどのように計算されて Scale に表示されるかは、図 15 と図 16 の例を示して解説します。図 15 と図 16 を見ると明らかなように、各 Item の横に表示されている数値は割合ではありません。これらの値の合計は 100 にはなりませんし、Scale での割合も、ラベル 25 で表される Item はラベル 20 の Item より 50%も大きくなっています。

例として、Linear と Circular スケールの両方で、最小値を 10、最大値を 110、その差を 100 とします。この場合、Scale 全体の領域は 100 ユニットになります。各 Item が領域に締める割合は、次の計算式で求めます。

$$\frac{ItemValue - ScaleMin}{ScaleMax - ScaleMin}$$

最小 Item の 20 は、上の式をあてはめると、
(20-10)/(110-10)=0.1=10%となります。また、最大 Item の 35 は、
(35-10)/(110-10)=2.5=25%となります。その他の Item を同様に計算すると、それぞれ、10%、15%、20%、25%となり、合計は 70%で、残りの 30%が図 15 の色の濃い部分として表示されます。一方、*clculation-by-ratio* モードでは、各 Item に割り当てられる領域は、次の計算式で求めます。

$$\frac{ItemValue - ScaleMin}{\sum_i (ItemValue_i - ScaleMin)}$$

Scale の最小値が 0 の時、両方の式では、Item の占める領域は、その値と比例することになります。*calculation-by-percent* モードでは、Item の値の合計と Scale の max-min 値の差が leftover の領域として表示されます。この合計が 100%より大きい場合、100%を超える部分の Item は表示されません。

6.2 Scale の選択

XmNxrtGaScale は、Circular Scale または Linear Scale のどちらかのオブジェクト・リファレンスを取ります。Percent ウィジェットに対して一つの Scale を作成する場合は、このオブジェクトが、使用される Scale として自動的に登録されます。二つ以上の Scale を作成する場合は、XtVaSetValues 内で Scale を渡すことにより、どちらの Scale を使用するか選択できます。次の例では、Circular Scale を作成し、Percent ウィジェットの XmNxrtGaScale に割り当てています。

```
Widget percent; /* A Percent widget */
Widget circular_scale; /* A Circular Scale object for the
                          percent widget. */
Widget linear_scale /* For later use */

circular_scale =
  XtVaCreateWidget("circular_scale",
      xmXrtGaCircularScaleObjectClass, percent,
      XmNxrtGaMin,                        DVAL(10.0),
      XmNxrtGaMax,                        DVAL(110.0),
      COLOR_RES(XmNbackground,          "midnightblue"),
      XmNxrtGaStartAngle,                 DVAL(0.0),
      XmNxrtGaStopAngle,                  DVAL(360.0),
```

```
        NULL);
scale = circular_scale;
XtVaSetValues(percent,
        XmNxrtGaScale,                          scale,
        NULL);
```

また、アプリケーションの別の場所で、XmNxrtGaScale に Linear Scale を割り当てることができます。XtVaSetValues を同様に呼び出して、今度はウィジェットが Linear Scale を表示するようにできます。

Scale は、ウィジェットのサイズのパラメータを定義します。この値は、XmNxrtGaMax でセットされる最大値と XmNxrtGaMin でセットされる最小値との差です。

XmNxrtGaItemList

Item リストは、Percent ウィジェットに Item が追加されるたびにアップデートされます。図 17 では、四つの Item がウィジェットに追加されています。ここでは、Circular Scale の開始値が 0、終了値が 100 となっており、Item の値の合計が 90 で、*calculation-by-percent* モードなので、leftover の部分が表示されています。これらの Item には全てラベルがついています。leftover を含む全ての Item は、Item リストによってアクセスできます。

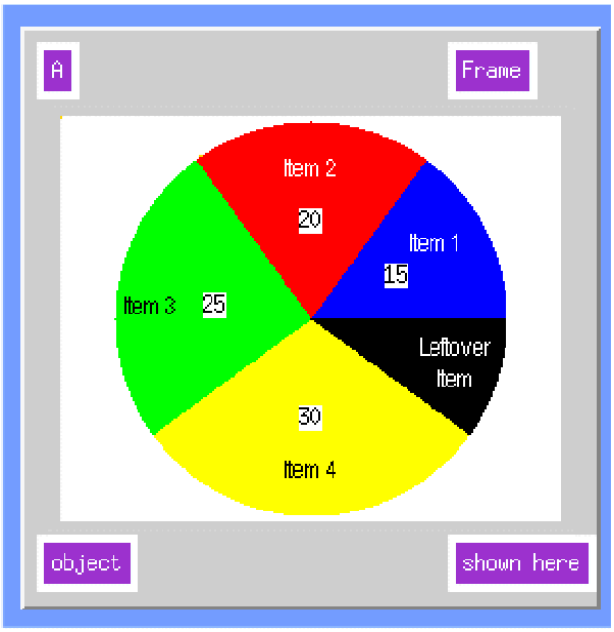


図17　フレーム、Item、テキストを表示している Percent ウィジェット

6.3　ラベル

各 Item のラベルの位置は、XmNxrtGaLabelExtent によってコントロールできます。

```
item = XtVaCreateWidget(MtCreateName(name, "item"),
  xmXrtGaItemObjectClass,     parent,
  XmNxrtGaValue,              DVAL(20.0),
  XmNxrtGaLabelExtent,        DVAL(0.5),
  NULL);
```

ラベルは、Crcular Scale 中央を基本にして表示されます。値 1.0 は、ラベルを円周上に配置します。Linear Scale では、縦方向の場合は左、横方向の場合は上部を基本にして表示されます。縦の Scale で、値が 0.25 の場合、ラベルは 25%左側から右寄りに配置されます。

ラベルの配置は Item オブジェクトに渡し、テキストは Text オブジェクトで管理させることもできます。

6.4 Item 上にラベルを配置

Text オブジェクトを使用して Item 上にラベルを配置することができます。Text オブジェクトは、下記のようにして XmNxrtGaValueObject により、オブジェクトを Item にセットします。

```
text =
  XtVaCreateWidget(MtCreateName(name, "text"),
      xmXrtGaTextObjectClass,              parent,
      XmNxrtGaString,                      "%g",
      XmNxrtGaIsShowing,                   True,
      XmNxrtGaValueObject,              item,
      XmNxrtGaAllowWidgetPositioning,   True,
      NULL);
```

ここでは、XmNxrtGaAllowWidgetPositioning が True になっています。この設定により、Item オブジェクトが位置を決めるようになります。値オブジェクトは、

```
      XmNxrtGaValueObject,              item,
```

により、Item オブジェクトにセットされています。これにより、Text オブジェクトが、item の XmNxrtGaValue リソースにアクセスして値に関する情報を取り出すようになります。その後、Text オブジェクトは、

```
      XmNxrtGaString,                      "%g",
```

の指定に従って、値をフォーマットします。上記の条件が揃うと、Text オブジェクトはフォーマットされた値を自動的に表示用テキストとみなします。ここでは、”%g”を XmNxrtGaString に指定していますので、Text オブジェクトは、ラベルに値をそのまま使用します。

6.5 Percent ウィジェットのリソースについての注意

Frame の配置

Percent ウィジェットには、XmNxrtGaFrame を使用して Frame を表示することができます。図 17 では、Percent ウィジェットは、灰色の長方形の部分です。その内側に配置されている白い Frame は、下記のようにマージンがセットされています。

```
frame =
  XtVaCreateWidget("frame",
      xmXrtGaFrameObjectClass,     percent,
      XmNxrtGaBottomMargin,           45,
      XmNxrtGaTopMargin,              45,
```

```
XmNxrtGaLeftMargin,              20,
XmNxrtGaRightMargin,         20,
COLOR_RES(XmNbackground,     "white"),
NULL);
```

XmNxrtGaTextList

Item オブジェクトにリストがあるのと同様に、Text オブジェクトにもリストがあります。図 17 では、四つの Text オブジェクトがウィジェットにセットされています。これらのオブジェクトも、リストとして管理され、XmNxrtGaTextList によってアクセスすることができます。

その他のリソース

Percent ウィジェットでは、XmNxrtGaImage を使用して背景にイメージをセットすることができます。このイメージは、XmNxrtGaImageLayout によって、中央かまたはタイルのように並べて配置するか指定できます。また、XmNxrtGaUseRenderList を True にセットし、XmNxrtGaRenderList によって指定された順序に Item が描画されるようにセットすることもできます。また、ウィジェットが描画されるタイミングも、XmNxrtGaRepaint を使用してコントロールすることができます。

6.6 トランスレーションとコールバック

Percent ウィジェットのトランスレーションは、下記の通りです。

```
CTRL+<Btn1Down>:  ZoomStart(Percent)
CTRL+<Btn1Motion>:    ZoomUpdate(Percent)
CTRL+<Btn1Up>:        ZoomEnd(Percent)
SHIFT+<Btn1Down>: TranslateStart(Percent)
SHIFT+<Btn1Motion>:TranslateUpdate(Percent)
SHIFT+<Btn1Up>:       TranslateEnd(Percent)
```

このウィジェットのコールバックは、XmNxrtGaExposeCallback、XmNxrtGaMapCallback、XmNxrtGaResizeCallback です。詳細は第 22 章を参照してください。

6.7 ウィジェットの概要

Include Files:	*<Xm/XrtGaPercent.h>*
Class Name:	XmXrtGaPercentWidget
Class Hierarchy:	Core ➔ XmPrimitive ➔ XmXrtGaPercentWidget
Class Pointer:	xmXrtGaPercentWidgetClass

6.8 リソースの一覧

Percent ウィジェットの全てのリソース及び、よく使用される継承されたリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

ウィジェットの背景色です。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_3D_OUT** **CSG**

ウィジェットの枠線の種類です。

XmNxrtGaCalcMode **XrtGaCalcMode** **XRTGA_CALC_MODE_PERCENT** **CSG**

各 Item の割合を計算する方法を指定します。有効な値は、XRTGA_CALC_MODE_PERCENT と XRTGA_CALC_MODE_RATIO です。Percent モードでは、*(value-min)*100/(max-min)*の式によって求められた、各 Item のパーセンテージが使用されます。Ratio モードでは、Item のパーセンテージの合計に対する各 Item の割合が使用されます。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDoubleBuffer **Boolean** **True** **CSG**

ウィジェットの描画をダブル・バッファリングするかどうか指定します。ダブル・バッファリングは、オブジェクトの描画、再描画時に生じるちらつきを現象させるテクニックです。

XmNxrtGaExposeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットが再描画したことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。XrtGaCallbackStruct のポインタが渡されます。

XmNxrtGaFrame **XrtGaObject** **NULL** **CSG**

ウィジェットの XrtGaFrame オブジェクト。Frame の種類は XmNxrtGaShape によって指定します。デフォルトでは、Frame は作成されていません。新しく Frame が作成されると、古い Frame と置き換わり、古い Frame はリストによって管理されるようになります。現在の Frame が破棄されると、このリソースは NULL にセットされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

ウィジェットの背景のイメージ。この値が NULL であれば、背景色が使用されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージを配置します。有効な値は、XRTGA_IMAGE_LAYOUT_CENTERED と XRTGA_IMAGE_LAYOUT_TILED です。

XmNxrtGaItemList **XrtGaList** ***dynamic*** **SG**

Percent ウィジェットが表示する Item のリストです。新しい Item オブジェクトは、親の Item リストに自動的に追加されます。このリストは、read-only で、その内容を操作することはできません。

XmNxrtGaMapCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットに *Map()*アクションが登録されたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。`XrtGaMapCallbackStruct` のポインタが各 map コールバックに渡されます。

XmNxrtGaRenderList **XrtGaList** ***dynamic*** **G**

`XmNxrtGaUseRenderList` が `True` であれば、ウィジェットは、`XmNxrtGaRenderList` に記憶されているリストに指定された順序に従って子オブジェクトを描画します。オブジェクトが新たに作成されると、このオブジェクトは自動的に render リストに追加されます。従って、デフォルトでは、render リストにある子オブジェクトの順序は、作成された順序と同じです。このリソースは read-only ですが、このリストのポインタを取得することにより、リストを直接操作することができます。

XmNxrtGaRepaint **Boolean** **True** **CSG**

ウィジェットのリソースが変更された時に再描画するかどうかを指定します。

XmNxrtGaResizeCallback **XtCallbackList** **NULL** **CSG**

XRT/gauge ウィジェットがリサイズされたことをアプリケーションに通知する時に呼び出されるコールバックのリストを指定します。`XrtGaCallbackStruct` のポインタが各コールバックに渡されます。

XmNxrtGaScale **XrtGaObject** **NULL** **SG**

Item の表示に必要な Circular または Linear Scale。この値が `NULL` の場合は、Item は表示されません。

XmNxrtGaTextList **XrtGaList** **NULL** **SG**

`XrtGaText` オブジェクトのリスト。Text オブジェクトを Item にセットするときは、Item を Text オブジェクトの `XmNxrtGaValueObject` リソースに渡してください。棒グラフでは、テキストは、Item の領域に中央揃えされます。円グラフでは、テキストはスライスの中央に配置されます。Circular Scale では、Item の `XmNxrtGaLabelExtent` が、Scale の半径に対するラベルの位置を決定し、Linear Scale では、ラベルの位置は Scale の幅と高さに対応します。

XmNxrtGaUseRenderList **Boolean** **False** **G**

このリソースの値が `True` の時、ウィジェットは `XmNxrtGaRenderList` に記憶されているリストに指定された順序で子オブジェクトを描画します。`False` の場合は、デフォルトの順序で描画されます。デフォルトでは、Frame、Circular または Linear スケール、Item、Item と関連している Text オブジェクト、Item と関連していない Text オブジェクトという順序に描画され、ほとんどの場合、デフォルトのままにしておくことができます。

6.9 手続きとマクロ

XmCreateXrtGaPercent()

Percent ウィジェットを作成します。

```
Widget
XmCreateXrtGaPercent(
  Widget      parent,
  String      name,
  Arg         *arglist,
  Int         argcount
  )
```

parent は親、*name* は作成されるウィジェットの名前です。*arglist* はウィジェットに適応される、*argcount* 引数のリストです。

XmIsXrtGaPercent()

```
XmIsXrtGaPercent(w)
```

`w` が `XrtGaPercent` ウィジェットまたは、`XrtGaPercent` のサブクラスであれば `True` を返すマクロです。

7

Center オブジェクト

XrtGaCenter ■ Center オブジェクトのリソースに関する注意
サンプル・コード ■ リソースの一覧
手続きと方法

7.1 XrtGaCenter

Center オブジェクトは、Circular Gauge ウィジェットに配置された Circular Scale の中央の位置をマークする時に使用されます。このオブジェクトには色をつけたり、独自のイメージを指定することができます。イメージが指定された場合は、背景は描画されません。また、イメージは、Center オブジェクトの半径に対してスケーリングされません。

Center オブジェクトは、Circular スケールにセットされていますが、`XmNxrtGaCircularScale` がセットされないと描画されません。

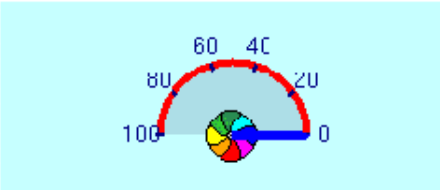


図18　Center オブジェクトとして使用されているイメージ

7.2 Center オブジェクトのリソースに関する注意

Center オブジェクトの色の設定

Center オブジェクトの色は、`XmNbackground` を使用してセットします。Scale の中央には、セットされた色の円形のディスクが表示されます。このディスクのサイズは、*radius* リソースでコントロールします。

Circular Scale との関連付け

Circular Gauge の各 Circular Scale オブジェクトは、一つまたは複数の Center オブジェクトを持っています。XmNxrtGaCircularScale を使用して、どの Circular Scale に Center オブジェクトを配置するか指定してください。

ディスクの大きさ

ディスクのサイズは XmNxrtGaRadius でセットします。ここで指定される値は、このオブジェクトがセットされる Circular Scale の半径に対応します。例えば、値 0.2 を指定すると、中央を示すディスクの半径は、Circular Scale の半径の 20%となります。

イメージの使用

Circular Scale の中央を示すときには、単なる円形のディスクのほかに、XmNxrtGaImage を使用して、イメージを使用することができます。詳細は、下記のサンプル･コードを参照してください。

表示のコントロール

Center オブジェクトの表示、非表示は XmNxrtGaIsShowing でコントロールしてください。

7.3 サンプル･コード

次のコードは、Circular スケールの中央に色付きのディスクを配置しています。

```
Widget circulargauge; /* A Circular Gauge widget */
/*Create Center*/
center = XtVaCreateWidget("center",
          xmXrtGaCenterObjectClass,    circulargauge,
          XmNxrtGaCircularScale,       scale,
          XmNxrtGaIsShowing,           True,
          XmNxrtGaRadius,              DVAL(0.1),
          COLOR_RES(XmNbackground,     "green"),
          NULL);
```

このディスクの代わりにイメージを使用する場合は、下記のようにしてください

```
char *name;
Widget circulargauge;
name = "$XRTHOME/src/icons/disk_3D_blue.xpm";
/*Create Center*/
      center = XtVaCreateWidget("center",
          xmXrtGaCenterObjectClass,    circulargauge,
          XmNxrtGaCircularScale,       scale,
          XmNxrtGaIsShowing,           True,
          XtVaTypedArg,                XmNxrtGaImage,
          XmRString,                   name,
          strlen(name) + 1,
          NULL);
```

XrtGaImage は、通常、各 XRT/gauge ウィジェットに登録されているイメージ･コンバータで作成されます。詳細は、第 23 章の XtConvertAndStore を使用したイメージの作成に関する説明を参照してください。

7.4 リソースの一覧

Center オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

オブジェクトの背景の色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaCircularScale **XrtGaObject** **NULL** **CSG**

オブジェクトがセットされる Circular Scale。この値が `NULL` の場合は、オブジェクトは描画されません。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースからの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

オブジェクトの背景に使用されるイメージです。このイメージは、それがセットされている Circular Scale の中央に表示されます。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Center オブジェクトを表示するかどうか指定します。

XmNxrtGaRadius **double** **1.0** **CSG**

Center オブジェクトのサイズを指定します。この値は、それがセットされている Circular Scale の `XmNxrtGaRadiusSize` の値に対応します。例えば、ここで値が 0.5 が指定されれば、Center オブジェクトの半径は、Scale の半径の半分ということになります。

7.5 手続きとマクロ

XmCreateXrtGaCenter()

`XrtGaCenter` オブジェクトを作成する、標準の Motif 形式の手続きです。

```
Widget
 XmCreateXrtGaCenter (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される `XrtGaCenter` の親、*name* は新しく作成された `XrtGaCenter` オブジェクトの名前です。*arglist* は、作成時に `XrtGaCenter` に渡される `XrtArgVal` リソースと値のペアのリストで、これは `NULL` とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaCenter オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaCenter()

```
XmIsXrtGaCenter(w)
```

w が XrtGaCenter オブジェクトまたは、XrtGaCenter オブジェクトのサブクラスの時は、True を返すマクロです。

8

Circular Scale オブジェクト

XrtGaCircularScale ■ Circular Scale の角度
Circular Scale のリソースについて ■ サンプル・コード
リソースの一覧 ■ 手続きと方法

8.1 XrtGaCircularScale

Circular Scale は、Circular Gauge ウィジェットや Circular Dial ウィジェットに使用されるオブジェクトです。Circular Dial は、`XmNxrtGaCircularScale` を使用して、Circular Scale を参照します。一方、Circular Gauge は、複数の Circular Scale を持つことができますので、`XmNxrtGaCircularScaleList` を使用して、これらのオブジェクトを参照します。

8.2 Circular Scale の角度

Circular Scale は、真東を 0 度として角度を測定します。角度は、反時計回りの方向に増加していきますので真北は 90 度、真西は 180 度、真南は 270 度となります。

図 19 は、x 軸や y 軸、0 度線の位置を表示し、どのように Circular Scale の角度が測定されるか示しています。

Circular Scale にマップされる最大と最小値とともに、開始と終了の角度が選択されると、もう一つのパラメータ、*direction* が、開始角度が最小値に対応するか、それとも最大値に対応するかを決定します。開始角度は、一般的には終了角度より小さくなりますが、この限りではありません。例えば、開始角度が 90 度、終了角度が 270 度で、測定される値が 100 から開始して 200 で終了する場合、*direction* を逆に設定すれば、値 200 は 90 度に位置することになり、値は Scale に沿って減少していき、値 100 にあたる 270 度で終了することになります。

角度を測定する座標システムは変更できませんが、Scale の値のパラメータが増加していく方向は逆にすることができます。

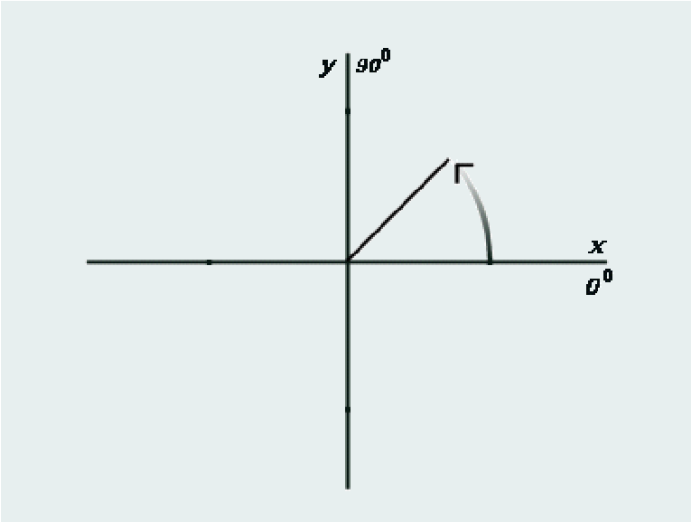


図19　Circular Scale の角度の測定法

8.3　Circular Scale のリソースについて

Circular Scale オブジェクトの役割は、Circular Gauge や Circular Dial を構築するときに必要な数値情報を示すことです。但し、このオブジェクトは、Scale の背景を表示するだけで、実際に情報を表示するのは、Needle や Tick、Range などです。図 20 には、これらのオブジェクトが全て表示されています。また、ウィジェット内における Scale の配置、サイズ、Range の値、表示される Scale の部分、Scale が示す値など、Circular Scale の幾つかのリソースもセットされています。

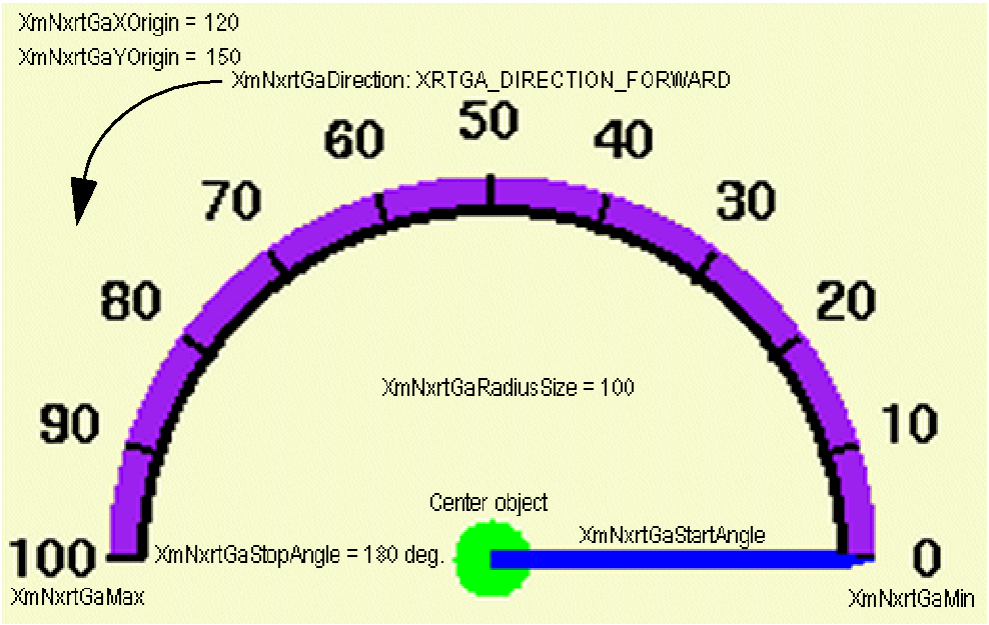


図20　各リソースがセットされた例を示すCircular Scale

その他に、エンドユーザがマウスを使用してウィジェット内の Scale を移動したりリサイズできるようにする XmNxrtGaAllowTransform というリソースがあります。このリソースがセットされると、XmNxrtGaTransformCallback が、トランスフォーメーションのプロセス中に呼び出されるコールバックのリストを指すようになります。Transform コールバックに関しては、トランスレーションとズームに関する解説を参照してください。

8.3.1 方向の指定

XmNxrtGaDirection は、Scale の値が増加する方向をコントロールします。このリソースは、次のどちらかにセットできます。

XRTGA_DIRECITON_FORWARD	値は反時計回りに増加
XRTGA_DIRECITON_BACKWARD	値は時計回りに増加

XmNxrtGaDirection のデフォルト値は XRTGA_DIRECITON_FORWARD です。

8.3.2 開始と終了の角度の指定

開始と終了の角度は、次の例のように、XmNxrtGaStartAngle と XmNxrtGaStopAngle でセットします。

```
double min = 0.0;
double max = 100.0;
 /*Create Basic Scale*/
 scale = XtVaCreateWidget("scale",
         xmXrtGaCircularScaleObjectClass,cgauge,
         XmNxrtGaStartAngle,      DVAL(0.0),
         XmNxrtGaStopAngle,       DVAL(180.0),
         XmNxrtGaMin,             DVAL(min),
         XmNxrtGaMax,             DVAL(max),
         NULL);
```

8.3.3 コールバック

このオブジェクトは、XmNxrtGaTransformCallback という transform コールバックをサポートしています。これらのコールバックについては、ユーザ・インタラクションのプログラミングに関する解説を参照してください。

8.4 サンプル・コード

次のコードは、Circular Scale を作成し、そのリソースを幾つかセットしています。

```
/*Create Basic Scale*/
scale = XtVaCreateWidget("scale",
         xmXrtGaCircularScaleObjectClass, cgauge,
         XmNxrtGaDirection,       XRTGA_DIRECTION_FORWARD,
         XmNxrtGaMax,             DVAL(100.0),
         XmNxrtGaMin,             DVAL(0.0),
         XmNxrtGaRadiusSize,      100,
         XmNxrtGaStartAngle,      DVAL(0.0),
         XmNxrtGaStopAngle,       DVAL(180.0),
         XmNxrtGaXOrigin,         0,
         XmNxrtGaYOrigin,         0,
         XmNxrtGaRadiusSize,      100,
```

```
                NULL);
    XtAddCallback(scale, XmNxrtGaTransformCallback, TransformCB, NULL);
```

8.5 リソースの一覧

Circular Scale オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

オブジェクトの背景の色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaAllowTransform **Boolean** **True** **CSG**

エンドユーザが Scale をズーム、移動できるかどうかを指定します。キーボードのアクションは、ズームが *Ctrl<Btn1Drag>*、移動が *Shift<Btn1Drag>*です。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースからの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDirection **XrtGaDirection** **XRTGA_DIRECTION_FORWARD** **CSG**

Scale が最小から最大へどちらの方向に測定されるか指定します。有効な値は、`XRTGA_DIRECTION_FORWARD` と `XRTGA_DIRECTION_BACKWARD` です。Forward は反時計回り、Backward は時計回りです。

XmNxrtGaMax **double** **1.0** **CSG**

Scale の最大値。

XmNxrtGaMin **double** **0.0** **CSG**

Scale の最小値。

XmNxrtGaRadiusSize **Dimension** ***dynamic*** **CSG**

Scale の半径です。デフォルトでは、親ウィジェットの幅と高さの小さいほうの三分の一です。

XmNxrtGaStartAngle **double** **0.0** **CSG**

Scale の開始角度。

XmNxrtGaStopAngle **double** **360.0** **CSG**

Scale の終了角度。

XmNxrtGaTransformCallback **XtCallbackList** **NULL** **CSG**

ユーザがオブジェクトとをズーム、移動すると呼び出されます。

XmNxrtGaXOrigin **Position** ***dynamic*** **CSG**

Scale の原点の x 座標。デフォルトは、親ウィジェットの中央です。

XmNxrtGaYOrigin **Position** ***dynamic*** **CSG**

Scale の原点の y 座標。デフォルトは親ウィジェットの中央です。

8.6 手続きとマクロ

Circular Scale の手続きと方法のリストです。

XmCreateXrtGaCircularScale()

```
Widget
 XmCreateXrtGaCircularScale (
         Widget      parent,
         char *      name,
         ArgList     arglist,
         Cardinal    argcount
         )
```

parent は作成される XrtGaCircularScale の親、*name* は新しく作成された XrtGaCircularScale オブジェクトの名前です。*arglist* は、作成時に XrtGaCircularScale に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaCircularScale オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaCircularScale()

```
XmIsXrtGaCircularScale(w)
```

w が XrtGaCircularScale オブジェクトまたは、XrtGaCircularScale オブジェクトのサブクラスの時は、True を返すマクロです。

9

Counter オブジェクト

XrtGaCounter ■ Counter リソース
コールバックとトランスレーション■サンプル・コード
ウィジェットの概要 ■ リソースの一覧
手続きと方法

9.1 XrtGaCounter

Counter オブジェクトは、Circular Gauge または Circular Dial に使用されます。

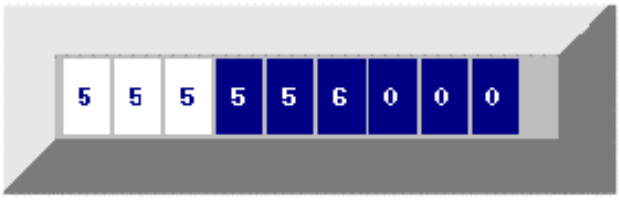


図21　Counter オブジェクト

9.2 Counter リソース

9.2.1 Counter オブジェクトをウィジェットとして使用する

Circular Gauge または Circular Dial を作成し、Counter オブジェクトをその内部に配置してください。ウィジェットの子としてその他のオブジェクトが作成されず、Counter オブジェクトが適当なサイズの場合、このウィジェットは、Counter のコンテナとしての役割を果たします。

9.2.2 枠線と背景の色の選択

Counter の周囲には枠線を表示できます。枠線の種類は、この章の最後のリソースの一覧を参照してください。

オブジェクトの背景色は、`XmNxrtGaSpacingHorizontal` または `XmNxrtGaSpacingVertical` が 0 でない場合に、各数値の周囲に表示されます。

9.2.3 色の選択

オブジェクト全体の背景色のほかに、Counter の各数値の部分にも前景と背景の色を指定することができます。前景色は数値表示に、背景色は数値の背景の矩形の部分に使用されます。

小数点の右側の桁では、背景色と前景色は逆になります。

9.2.4 フォントの選択

フォントは、通常、次のようにして fallback 配列でセットされます。

```
static String fallbacks[] = {
  "*.counter.fontList:
  -*-times-medium-r-normal--24-140-*-*-*-*-*-*",
  NULL
  };
```

XtVaTypedArg を使用してもセットすることができます。

9.2.5 背景にイメージを使用

オブジェクトの背景には、イメージを表示させることができます。このイメージは、オブジェクトの中央に配置したり、タイルのように並べることができます。

9.2.6 小数点の設定

Counter オブジェクトは、小数点のアイコンを持っていません。小数点を境にして右側と左側の桁の前景と背景色を入れ替えて小数点数値を表示します。アプリケーションで小数点が必要な場合は、LEDNumerical ウィジェットを使用してください。

9.2.7 Transitions の使用

XmNxrtGaUseTransition を True にして、数値が次の数値にロールオーバーされるように設定することができます。この設定により、Counter オブジェクトは、0 から 9 へ数値がスムーズに移り変わっていく数値カウンタのような動作をします。移り変わりの速度は、XmNxrtGaTransitonRate でコントロールします。ここで設定される値は、ミリ秒で表された、ひとつのフレーム(数値の表示)から次への移動にかかる時間です。フレームの数は XmNxrtGaTransitionsSteps でセットしてください。

9.2.8 リセット・ボタン

リセット・ボタンをつけることにより、ユーザはカウンタをリセットすることができます。このボタンは、カウンタの数値表示の前か後に配置され、矩形、円形の形のほかに独自のイメージを表示させることができます。

リセット・ボタンのサイズは、高さや幅、オフセットをセットして指定します。左と右のオフセットは、XmNxrtGaResetOffset リソースによってコントロールされ、リセット・ボタンは、数値と並んで表示されます。

9.2.9 Counter への値の設定

表示される値は、XmNxrtGaValue によって Counter に指定してください。この時、アプリケーションが必要とする数値に十分な数の桁数が用意されるように、XmNxrtGaNumDigits がセットされるようにしてください。

9.3 コールバックとトランスレーション

コールバック

カウンタの数値の変更に対応するには、XmNxrtGaValueChangedCallback を使用してください。詳細は第 22 章を参照してください。

トレンスレーション

リセット・ボタンが表示されている場合、ウィジェットは自動的に *ResetCounter()* を *<Btn1Down>* と *<Btn1Up>* イベントに登録します。ユーザによってリセット・ボタンが押されると、*ResetCounter()* は、(*<Btn1Down>* または *<Btn1Up>* によって)XRTGA_REASON_RESET_COUNTER_BEGIN または XRTGA_RESET_COUNTER_END の *reason* の Interaction コールバックを呼び出します。リセットするには、同じカウンタの *<btn1down>* と *<btn1up>* を押すようにしてください。このコールバックを呼び出すことにより、ユーザは、対応する Needle や Switch のインタラクションを不可にすることができます。

9.4 サンプル・コード

下記のサンプル・コードは、Counter オブジェクトを作成し、幾つかのリソースをセットしています。

```
counter =
XtVaCreateWidget("counter",
     xmXrtGaCounterObjectClass,       cgauge,
     XmNbackground,                   COLOR_RES("grey"),
     XmNxrtGaBorder,                  XRTGA_BORDER_NONE,
     XmNxrtGaBorderWidth,             (Dimension) 2,
     XmNxrtGaDebug,                   True,
     XmNxrtGaDecimalPlaces,           2,
     XmNxrtGaDigitBackgroundColor,    COLOR_RES("red"),
     XmNxrtGaDigitForegroundColor,    COLOR_RES("blue"),
     XmNxrtGaDoubleBuffer,            True,
     XmNxrtGaImage,                   NULL,
     XmNxrtGaImageLayout,             XRTGA_IMAGE_LAYOUT_TILED,
     XmNxrtGaIsShowing,               True,
     XmNxrtGaNumDigits,               7,
     XmNxrtGaRepaint,                 True,
     XmNxrtGaResetAlignment,          XRTGA_ALIGNMENT_END,
     XmNxrtGaResetBackgroundColor,COLOR_RES("green"),
     XmNxrtGaResetHeight,             (Dimension) 10,
     XmNxrtGaResetImage,              NULL,
     XmNxrtGaResetOffset,             (Dimension) 2,
     XmNxrtGaResetShape,              XRTGA_SHAPE_CIRCLE,
```

```
        XmNxrtGaResetShow,              True,
        XmNxrtGaResetWidth,             (Dimension) 50,
        XmNxrtGaSpacingHorizontal,      (Dimension) 15,
        XmNxrtGaSpacingVertical,        (Dimension) 5,
        XmNxrtGaTransitionRate,         100,
        XmNxrtGaTransitionSteps,        10,
        XmNxrtGaUseTransition,          True,
        XmNxrtGaValue,                  DVAL(12345.67),
        NULL);
    XtAddCallback(counter, XmNxrtGaValueChangedCallback,
        (XtCallbackProc)valueChangedCB, NULL);
```

9.5 ウィジェットの概要

Include Files: *<Xm/XrtGaCounter.h>*

Class Name: XmXrtGaCounter

Class Hierarchy: Core ➔ XmPrimitive ➔ XmXrtGaCounter

Class Pointer: xmXrtGaCounterWidgetClass

9.6 リソースの一覧

Counter オブジェクトの全てのリソース及び、よく使用される継承されたリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

オブジェクトの背景色です。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNfontList **XmFontList** **“fixed”** **CSG**

オブジェクトに表示されるテキストのフォント。

XmNheight **Dimension** ***dynamic*** **CSG**

オブジェクトの高さ。

XmNwidth **Dimension** ***dynamic*** **CSG**

オブジェクトの幅

XmNx **Position** **0** **CSG**

Pixel で表される、親ウィジェットに対するオブジェクトの左上角の横方向の位置。

XmNy **Position** **0** **CSG**

Pixel で表される、親ウィジェットに対するオブジェクトの左上角の縦方向の位置。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_3D_OUT** **CSG**

オブジェクトの罫線。種類は次の通りです。

XRTGA_BORDER_NONE	罫線なし
XRTGA_BORDER_PLAIN	前景色で描かれる装飾無しの単線
XRTGA_BORDER_3D_IN	オブジェクトが沈んだようにみえる線
XRTGA_BORDER_3D_OUT	オブジェクトが浮き上がったようにみえる線
XRTGA_BORDER_ETCHED_IN	オブジェクトが沈んだようにみえる二重線
XRTGA_BORDER_ETCHED_OUT	オブジェクトが浮き上がったようにみえる二重線
XRTGA_BORDER_FRAME_IN	オブジェクトが沈んだように見えるフレーム
XRTGA_BORDER_FRAME_OUT	オブジェクトが浮き上がったようにフレーム
XRTGA_BORDER_BEVEL	オブジェクトの周囲に描かれる枠線
XRTGA_BORDER_SHADOW	オブジェクトの下に描画される影

XmNxrtGaBorderWidth **Dimension** **2** **CSG**

オブジェクトの枠線の幅を pixel で指定。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDecimalPlaces **int** **0** **CSG**

表示する小数点以下の桁。負の数が指定されると 0 とみなされ、小数点は表示されません。小数点より右側の桁の色は反転されます。

XmNxrtGaDigitBackgroundColor **Pixel** **inherited** **CSG**

小数点より左側の桁の背景色、及び、右側の桁の前景色。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaDigitForegroundColor **Pixel** **inherited** **CSG**

小数点より左側の桁の前景色、及び、右側の桁の背景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

オブジェクトの背景のイメージ。値が NULL の場合は、背景色が使用されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージの配置を指定します。有効な値は、XRTGA_IMAGE_LAYOUT_CENTERED と XRTGA_IMAGE_LAYOUT_TILED です。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

オブジェクトの表示、非表示を指定します。

XmNxrtGaNumDigits **int** **5** **CSG**

表示される桁数。先頭のプラスやマイナスの記号は含まれません。

XmNxrtGaResetAlignment **XrtGaAlignment** **XRTGA_ALIGNMENT_END** **CSG**

XrtGaCounter オブジェクト内にどのようにリセットのアイコンを配置するか指定します。有効な値は XRTGA_ALIGNMENT_BEGIN と XRTGA_ALIGNMENT_END です。XRTGA_ALIGNMENT_CENTER は無視されます。

XmNxrtGaResetBackgroundColor **Pixel** ***dynamic*** **CSG**

リセット・ボタンの背景色です。このリソースがセットされない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaResetBackgroundColor **Pixel** ***inherited*** **CSG**

Counter オブジェクト内のリセット・ボタンの背景色。

XmNxrtGaResetImage **XrtGaResetImage** **NULL** **CSG**

リセット・ボタンの背景のイメージ。イメージは中央に配置されます。

XmNxrtGaResetOffset **Dimension** **0** **CSG**

リセット・アイコンの横方向のオフセットを指定します。この値は、リセット・ボタンの左と右側に適用されます。

XmNxrtGaResetShape **XrtGaResetShape** **XRTGA_SHAPE_RECTANGLE** **CSG**

リセット・ボタンの形です。矩形または円形の場合、オンとオフの色がその形を塗りつぶします。このリソースには温度計やタンクの形や使用できません。有効な値は次の通りです。

XRTGA_SHAPE_RECTANGLE	矩形
XRTGA_SHAPE_CIRCLE	円形
XRTGA_SHAPE_IMAGE	XmNxrtGaResetImage に指定された XrtGaResetImage が使用される

XmNxrtGaResetShow **Boolean** **False** **CSG**

True の場合、リセット・ボタンは、数値の左側に表示されます。このボタンをクリックすると、値は 0 にリセットされます。

XmNxrtGaResetWidth **Dimension** ***dynamic*** **CSG**

Counter オブジェクトのリセット・アイコンの幅。

XmNxrtGaSpacingHorizontal **Dimension** **2** **CSG**

Pixel で指定される桁の間のスペース。

XmNxrtGaSpacingVertical **Dimension** **5** **CSG**

Pixel で指定される数値の上下のスペース。

XmNxrtGaTransitionRate **int** **10** **CSG**

ミリ秒で指定されるカウンタの数値が移り変わる速さ。

XmNxrtGaTransitionSteps **int** **10** **CSG**

現在の値から次の値の間に入るステップ(移り変わる数値)の数。

XmNxrtGaUseTransition **Boolean** **False** **CSG**

`True` の場合、transition の速さとステップが使用されます。`False` の場合は、数値は瞬時に値を表示します。

XmNxrtGaValue **double** **0.0** **CSG**

Counter の値。

XmNxrtGaValueChangedCallback **XtCallbackList** **NULL** **CSG**

値が変更された時に呼び出されるコールバックのリスト。`XmNxrtGaValueChangedCallbackStruct` のポインタが各コールバックに渡されます。

9.7 手続きとマクロ

XmCreateXrtGaCounter()

```
Widget
XmCreateXrtGaCounter(
      Widget   parent,
      Char *   name,
      Arg      *arglist,
      Cardinal argcount
      )
```

parent は親、*name* は作成されるオブジェクトの名前です。*arglist* は *argcount* 引数のリストです。

XmIsXrtGaCounter()

```
XmIsXrtGaCounter(w)
```

`w` が `XrtGaCounter` オブジェクトまたは、`XrtGaCounter` のサブクラスであれば `True` を返すマクロです。

10

Frame オブジェクト

XrtGaFrame ■ Frame オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

10.1 XrtGaFrame

Frame は、Circular Gauge、Circular Dial、Linear Gauge、Linear Dial、Percent ウィジェットに使用されます。このオブジェクトは、これらのウィジェットの内側に背景色で描画された部分を表示します。

左右、上下のマージンが Frame の位置をセットします。ウィジェットのデフォルトの描画の順序では、Frame は背景色の直後に描画されます。

`XrtGaFrame` オブジェクトの形は、`XmNxrtGaShape` でコントロールします。形は、矩形(デフォルト)、円形のほか、イメージを選択することができます。円形の場合、指定されたサイズに合うように楕円形で表示されます。

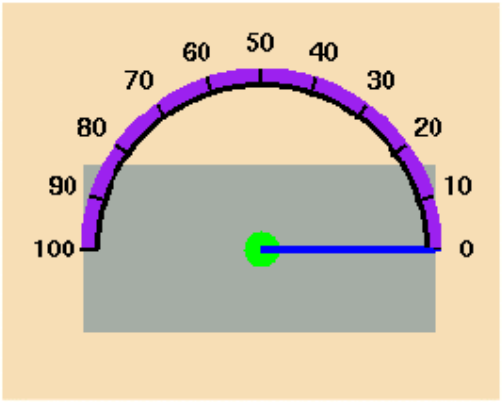


図22　マージンを調整して、ウィジェットの内側に収まった灰色の Frame

親ウイジェットには、デフォルトでは Frame オブジェクトは作成されません。新しい Frame が作成されると、既存のものと置き換わり、古い Frame は、ウィジェットによって管理される Frame のリストで管理されるようになります。

アクティブな Frame が破棄されると、ウィジェットの XmNxrtGaFrame は NULL にセットされます。

10.2 Frame オブジェクトのリソースについて

Frame の色の設定には、XmNbackground を使用します。

デバッグに関する情報を得る場合には、XmNxrtGaDebug を True にしてください。

Frame を非表示にするには、XmNxrtGaIsShowing を False にしてください。

Frame の種類は XmNxrtGaShape を使用してセットします。矩形や円形の他、独自に指定したイメージを使用することができます。

Frame の位置は XmNxrtGaBottomMargin 、 XmNxrtGaLeftMargin 、 XmNxrtGaRightMargin、XmNxrtGaTopMargin でコントロールします。これらのマージンの値は、pixel で表されます。ここで指定されるマージンは、XmNxrtGaHighlightThickness で指定された pixel 値からカウントされます。ウィジェットの枠線も、この部分から描画が開始されます。

10.3 サンプル･コード

下記のコードでは、Frame の種類、色、各マージンの値をセットしています。

```
Widget cgauge; /* The parent widget - a circular gauge */
Widget cgauge_frame; /* The frame object */
/* Create frame for a Circular Gauge */
cgauge_frame =
 XtVaCreateWidget("cgauge_frame",
     xmXrtGaFrameObjectClass,     cgauge,
     XmNxrtGaFrameType,           XRTGA_SHAPE_CIRCLE,
     XmNxrtGaIsShowing,           True,
     XmNxrtGaTopMargin,           2,
     XmNxrtGaLeftMargin,          3,
     XmNxrtGaRightMargin,         4,
     XmNxrtGaBottomMargin,        5,
     COLOR_RES(XmNbackground,     "white"),
     NULL);
```

10.4 リソースの一覧

Frame オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

Frame の背景の色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaBottomMargin **int** **0** **CSG**

下部のマージン。Frame のマージンは、ウィジェットの端からではなく、ウィジェットの枠線の XmNhighlightThickness から開始されます。ウィジェットの枠線もここから開始されます。枠線は、通常、Frame 上に描画されますので、マージンが表示されるようにするには、マージンを枠線幅より大きく設定してください。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Frame オブジェクトの背景のイメージ。イメージは、Frame の中央に配置されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージの配置を指定します。有効な値は、XRTGA_IMAGE_LAYOUT_CENTERED と XRTGA_IMAGE_LAYOUT_TILED です。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Frame の表示、非表示を指定します。

XmNxrtGaLeftMargin **int** **0** **CSG**

左側のマージン。Frame のマージンは、ウィジェットの端からではなく、ウィジェットの枠線の XmNhighlightThickness から開始されます。ウィジェットの枠線もここから開始されます。枠線は、通常、Frame 上に描画されますので、マージンが表示されるようにするには、マージンを枠線幅より大きく設定してください。

XmNxrtGaRightMargin **int** **0** **CSG**

右側のマージン。Frame のマージンは、ウィジェットの端からではなく、ウィジェットの枠線の XmNhighlightThickness から開始されます。ウィジェットの枠線もここから開始されます。枠線は、通常、Frame 上に描画されますので、マージンが表示されるようにするには、マージンを枠線幅より大きく設定してください。

XmNxrtGaShape **XrtGaShape** **XRTGA_SHAPE_RECTANGLE** **CSG**

Frame の形。有効な値は、XRTGA_SHAPE_RECTANGLE、XRTGA_SHAPE_CIRCLE、XRTGA_SHAPE_IMAGE です。

XmNxrtGaTopMargin	int	0	CSG

上部のマージン。Frame のマージンは、ウィジェットの端からではなく、ウィジェットの枠線の XmNhighlightThickness から開始されます。ウィジェットの枠線もここから開始されます。枠線は、通常、Frame 上に描画されますので、マージンが表示されるようにするには、マージンを枠線幅より大きく設定してください。

10.5 手続きとマクロ

XmCreateXrtGaFrame()

XrtGaFrame オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaFrame (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaFrame の親、*name* は新しく作成された XrtGaFrame オブジェクトの名前です。*arglist* は、作成時に XrtGaFrame に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaFrame オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaFrame()

```
XmIsXrtGaFrame(w)
```

w が XrtGaFrame オブジェクトまたは、XrtGaFrame オブジェクトのサブクラスの時は、True を返すマクロです。

11

Icon オブジェクト

XrtGaIcon ■ Icon のリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

11.1 XrtGaIcon

Icon オブジェクトは、LEDIcon ウィジェットと一緒に使用されます。

LEDIcon ウィジェットは、プロセス等の進行具合を表示する時に必要な数の Icon で構成されます。Icon は、この Scale 上での現在値を表示していくマーカの役割を持ちます。ウィジェットが XRTGA_DISPLAY_MODE_MAGNITUDE モードの場合、現在値より小さい値の Icon はオンの色で表示され、それより大きい値の Icon はオフの色で表示されます。ウィジェットが XRTGA_DISPLAY_MODE_BINARY の場合は、バイナリ値を表す Icon はオンの色、バイナリ 0 の Icon はオフの色で表示されます。

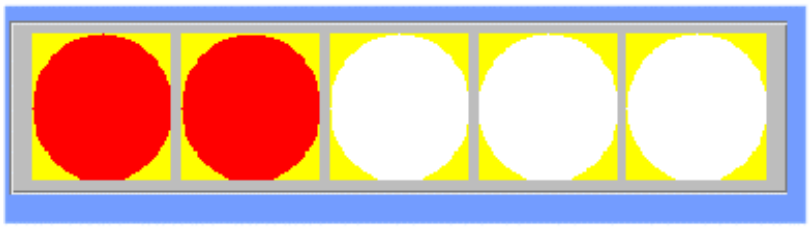


図23　オフの色が白の円形の Icon

11.2 Icon のリソースについて

11.2.1 Icon の色

Icon には三つの色が使用できます。XmNbackground は Icon の背景の色を指定します。デフォルトでは、Icon の矩形がその領域を全て塗りつぶしますので、背景色は見えません。

背景色は、Icon の形が円形や、Icon 全体を覆ってしまわないイメージが使用された時に表示されます。

Icon のオンとオフの色は、XmNxrtGaOnColor と XmNxrtGaOffColor を使用してセットしてください。

11.2.2 Icon にイメージを使用する

XmNxrtGaOnImage と XmNxrtGaOffImage で Icon のオンとオフのイメージを指定します。イメージは、Icon の中央に配置され、はみ出た部分は切り取られます。

11.2.3 Icon の形

Icon には矩形か円形の形を指定できます。円形は、Icon の矩形の境界線の内側に描かれます。Icon の形が矩形の場合、楕円が描かれます。これらの形のほかに、XmNxrtGaShape を使用してイメージを指定することもできます。

Icon では、温度計やタンクの形は使用できません。

11.3 サンプル・コード

```
Widget led; /* The parent LEDIcon widget */

/* Create an icon and add it to the icon list */
XtVaCreateWidget(icon_name,
      xmXrtGaIconObjectClass,      led,
      COLOR_RES(XmNbackground,     "yellow",
      COLOR_RES(XmNxrtGaOnColor,   "red",
      COLOR_RES(XmNxrtGaOffColor,  "white",
      XmNxrtGaShape,               XRTGA_SHAPE_CIRCLE,
      NULL);
```

図 23 の Icon は、上記のコードによって作成されています。

XmNxrtGaOnImage と XmNxrtGaOffImage を使用して、オンとオフのイメージを表示させることができます。

11.4 リソースの一覧

Icon オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

Icon オブジェクトの背景の色です。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaOffColor **Pixel** ***inherited*** **CSG**

選択されていない状態を示す色。Icon がオフの場合、指定された領域がこの色で塗りつぶされます。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaOnColor **Pixel** ***inherited*** **CSG**

選択された状態を示す色。Icon がオンの場合、指定された領域がこの色で塗りつぶされます。このリソースがセットされていない場合は、親ウィジェットの背景色が使用されます。

XmNxrtGaOnImage **XrtGaImage** **NULL** **CSG**

選択された状態の Icon のイメージ。このイメージは、Icon 内に中央に配置されます。Icon のサイズは、その親によって決定されます。Icon がイメージより小さい場合、はみ出た部分は切り取られます。

XmNxrtGaOffImage **XrtGaImage** **NULL** **CSG**

選択されていない状態の Icon のイメージ。このイメージは、Icon 内に中央に配置されます。Icon のサイズは、その親によって決定されます。Icon がイメージより小さい場合、はみ出た部分は切り取られます。

XmNxrtGaShape **XrtGaShape** **XRTGA_SHAPE_RECTANGLE CSG** **CSG**

Icon の形。XRTGA_SHAPE_RECTANGLE または XRTGA_SHAPE_CIRCLE の場合、オンとオフの色が指定された領域を塗りつぶし、XRTGA_SHAPE_IMAGE の場合はイメージが表示されます。このリソースには、温度計やタンクの形を指定することはできません。Icon のサイズは、親のサイズに依存します。

11.5 手続きとマクロ

XmCreateXrtGaIcon()

XrtGaIcon オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaIcon (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaIcon の親、*name* は新しく作成された XrtGaIcon オブジェクトの名前です。*arglist* は、作成時に XrtGaIcon に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaIcon オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaIcon()

```
XmIsXrtGaIcon(w)
```

wがXrtGaIconオブジェクトまたは、XrtGaIconオブジェクトのサブクラスの時は、True を返すマクロです。

12

Indicator オブジェクト

XrtGaIndicator ■ Indicator のリソースについて
コールバック ■ サンプル・コード
リソースの一覧 ■ 手続きと方法

12.1 XrtGaIndicator

Indicator は、Linear Dial のオンとオフの色(またはイメージ)の境目や、Linear Gauge の min と max の間の任意の場所へマークとして表示されるオブジェクトです。Indicator を表示するには、Indicator が表示される Scale がセットされている必要があります。Linear Dial の場合は、アクティブな Indicator は一つしか使用できません。

一方、Linear Gauge には複数の Indicator を表示させることができます。Linear Dial では、デフォルトで一つの Indicator が親ウィジェットに対して作成されます。`XmNxrtGaIndicator` がデフォルト値になっている(即ち、デフォルトの Indicator を指している)時に新しい Indicator が作成されると、このリソースは自動的に新しいオブジェクトを指すようになり、デフォルトの Indicator は破棄されます。もし、このリソースが指している Indicator が破棄されれば、このリソースは `NULL` にセットされます。

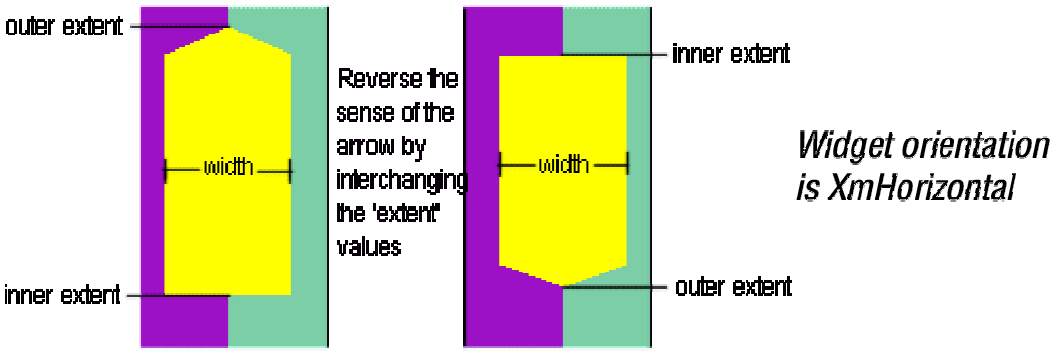


図24　ポインタの形をした Indicator

12.2 Indicator のリソースについて

12.2.1 Indicator のリソース

デバッグに関する情報については、XmNxrtGaDebug を使用してください。

任意の Indicator を表示するには、独自の描画ファンクションを使用し、XmNxrtGaDrawFunc で参照するようにしてください。詳細は、第 23 章の”独自の描画ファンクションの使用”を参照してください。

XmNxrtGaImage を使用して、Indicator には、イメージを指定することができます。親の方向が変化したり、outer extent が inner extent より大きくなった(ポインタの方向が変更した)ときに、イメージの方向も変更されるようにする場合は、XmNxrtGaImageRotate を True にしてください。

12.2.2 Indicator のサイズ

XmNxrtGaInnerExtent と XmNxrtGaOuterExtent で Indicator の長さをどこからどこまでとするか、XmNwidth で Indicator の幅を設定します。両方のサイズ決めのリソースは、それがセットされる Linear Scale のサイズに対する割合をパラメータとして取ります。図 24 の Linear Scale の Indicator では、inner extent は 0.8、outer extent は 0.1 になっています。

12.2.3 ポインタやトライアングルの方向を逆にする

ポインタやトライアングルの方向を逆にする場合は、XmNxrtGaInnerExtent と XmNxrtGaOuterExtent を入れ替えてください。これにより、元の形と全く同じサイズで逆のイメージが作成できます。

12.3 コールバック

このオブジェクトに定義されているコールバックは XmNxrtGaValueChangedCallback です。これは、Indicator の値が変更された時に呼び出されるコールバックのリストです。XrtGaValueChangedCallbackStruct のポインタが各コールバックに渡されます。詳細は、第 22 章を参照してください。

12.4 サンプル・コード

下記のコードは、Scale 上で値 20 の場所に青い Indicator を配置しています。

```
Widget lgauge; /* The parent linear gauge */
Widget scale; /* A scale associated with the linear gauge*/
XrtGaIndicatorType indicator_type;

/*Create an indicator which will remain at the daily maximum value*/
 indicator_type = XRTGA_INDICATOR_TYPE_POINTER;
 max_indicator =
```

```
XtVaCreateWidget("max_indicator",
        xmXrtGaIndicatorObjectClass, lgauge,
        XmNxrtGaLinearScale,         scale,
        XmNbackground,               COLOR_RES("grey"),
        XmNwidth,                    5,
        XmNxrtGaValue,               DVAL(20.0),
        XmNxrtGaIndicatorType,       indicator_type,
        XmNxrtGaInnerExtent,         DVAL(0.20),
        XmNxrtGaOuterExtent,         DVAL(0.38),
        COLOR_RES(XmNbackground,     "blue"),
        XmNxrtGaIsShowing,           True,
        NULL);
```

12.5 リソースの一覧

Indicator オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

オブジェクトの背景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNwidth **Dimension** **0** **CSG**

Indicator の最も広い幅(pixel)。円形やイメージが使用されている場合には、このリソースは無視されます。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDrawFunc **XrtGaDrawFunc** **NULL** **CSG**

オブジェクトの描画ファンクション。この値が `NULL` でなければ、このファンクションが Indicator の描画に使用されます。`XmNxrtGaIndicatorType` は無視されます。詳細は、P181 を参照してください。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Indicator に使用されるイメージ。

XmNxrtGaImageRotate **Boolean** **True** **CSG**

親ウィジェットの方向や、(`XmNxrtGaInnerExtent` と `XmNxrtGaOuterExtent` で表される)ポインタの方向が変更された時に、イメージの方向も変更するかどうか指定します。

XmNxrtGaInnerExtent **double** **0.0** **CSG**

Indicator オブジェクトの inner extent。横方向のスケールでは、この値が 0.2 であると、Scale の高さの 0.2 倍分だけ、Scale の上部から離れていることを示します。Scale の位置は、Scale の左上の座標値を決定します。

XmNxrtGaInteractionType **XrtGaInteractionType** **XRTGA_INTERACTION_TYPE_NONE** **CSG**

Indicaotr のマウスのインタラクションの型を指定します。有効な値は、`XRTGA_INTERACTION_TYPE_NONE`、`XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE`、`XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE` です。

XmNxrtGaIndicatorType **XrtGaIndicatorType** **XRTGA_INDICATOR_TYPE_RECTANGLE** **CSG**

Indicator の型を指定します。有効な値は、`XRTGA_INDICATOR_TYPE_RECTANGLE`、`XRTGA_INDICATOR_TYPE_CIRCLE`、`XRTGA_INDICATOR_TYPE_POINTER`、`XRTGA_INDICATOR_TYPE_TRIANGLE`、`XRTGA_INDICATOR_TYPE_IMAGE` です。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

オブジェクトの表示、非表示を指定します。

XmNxrtGaLinearScale **XrtGaObject** **NULL** **CSG**

オブジェクトがセットされる Linear Scale。この値が `NULL` の場合は、Indicator は表示されません。

XmNxrtGaOuterExtent **double** **1.0** **CSG**

Indicator オブジェクトの outer extent。横方向のスケールでは、この値が 0.8 であると、Scale の高さの 0.8 倍分だけ、Scale の上部から離れていることを示します。Scale の位置は、Scale の左上の座標値を決定します。

XmNxrtGaValue **double** ***dynamic*** **CSG**

Indicator の現在の値。この値は、Indicator がセットされている Scale の *min* と *max* 値の間でなければなりません。最大値より大きい値は *max* の位置に、最小値より小さい値は *min* の位置に表示されてしまいます。

XmNxrtGaValueChangedCallback **XtCallbackList** **NULL** **CSG**

Indicator の値が変更された時に呼び出されるコールバックのリスト。`XrtGaValueChangedCallbackStruct` のポインタが各コールバックに渡されます。

12.6 手続きとマクロ

XmCreateXrtGaIndicator()

XrtGaIndicator オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaIndicator (
     Widget       parent,
     char *       name,
     ArgList      arglist,
     Cardinal     argcount
     )
```

parent は作成される XrtGaIndicator の親、*name* は新しく作成された XrtGaIndicator オブジェクトの名前です。*arglist* は、作成時に XrtGaIndicator に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaIndicator オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaIndicator()

```
XmIsXrtGaIndicator(w)
```

w が XrtGaIndicator オブジェクトまたは、XrtGaIndicator オブジェクトのサブクラスの時は、True を返すマクロです。

13

Item オブジェクト

XrtGaItem ■ Item オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

13.1 XrtGaItem

Percent ウィジェットは *Item* で構成されています。各 Item は数や量を表し、これらをまとめて、ある物質や状態などがどのように構成されているかを示すことができます。例えば、ポートフォリオを構成する株や債権の価格、化合物を構成するエレメントの重さ、企業の各部署ごとの従業員の数などは、Item を使用することにより、各構成要素が全体のどの程度になるか示すことができます。各 Item はそれぞれ値を持ちますが、その合計は、Percent ウィジェットにセットされている値より小さくなければなりません。

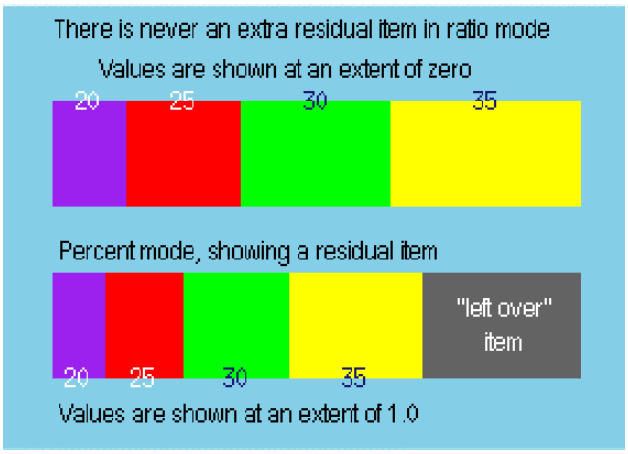


図25　ratio と percent モードで表示された Percent ウィジェット内の Item

図 25 の例のように、percent モードの場合は、Percent ウィジェットに"unnamed"Item を表示させることができます。この Item は、それがセットされる Circular または Linear Scale の max-min 値より、名前がついている Item の合計の値が小さい場合に表示されます。

13.2 Item オブジェクトのリソースについて

イメージの使用

Item にイメージをセットするには、XmNxrtGaImage を使用してください。イメージは、Item の領域の中央に配置されます。このリソースは、Linear Scale の場合にのみ使用できます。

ラベル

Item には Text オブジェクトを使用してラベルを貼り付けることができます。Text オブジェクトの XmNxrtGaValueObject は、次のようにして Item にセットできます。

```
text =
 XtVaCreateWidget(MtCreateName(name, "text"),
     xmXrtGaTextObjectClass,          parent,
     XmNxrtGaString,                  "%g",
     XmNxrtGaIsShowing,               True,
     XmNxrtGaValueObject,             item,
     XmNxrtGaAllowWidgetPositioning,  True,
     NULL);
```

ここでは、XmNxrtGaAllowWidgetPositioning が True になっています。この設定により、Item オブジェクトが位置決めをするようになります。また、XmNxrtGaValueObject は

```
     XmNxrtGaValueObject,             item,
```

とセットされています。これは、Text オブジェクトに、Item の XmNxrtGaValue リソースにアクセスするように指示しています。XmNxrtGaString は次のようにしてフォーマットを指定しています。

```
     XmNxrtGaString,                  "%g",
```

これらの設定が全て正しく行われれば、Text オブジェクトは、Item のフォーマットされた値を表示テキストとしてみなします。ここでは、"%g"とあるので、Text オブジェクトは Item の値をラベルとして表示します。

ラベルの位置

ラベルの位置は XmNxrtGaLabelExtent を使用して指定します。このリソースは、Item がセットされる Circular または Linear Scale に対する割合をパラメータとして取ります。図 25 は、二つの Linear Scale を表示しており、上の Scale のパラメータは 0 なのでラベルは Scale の上部、下の Scale のパラメータは 1.0 なのでラベルは Scale の下部に表示されています。

Circular Scale では、円の中央が開始点、円周上が 1.0 となります。

Item の値

Item の値は XmNxrtGaValue を使用してセットしてください。Percent ウィジェットが *calculation-by-ratio* モードの場合、Item の値の合計が計算され、その合計に基づいて、各 Item が全体に占める割合が計算されます。Percent ウィジェットが使用する Scale には合計値をセットすることができます。

calculation-by-percent モードの場合、全ての Item の合計がその合計値より小さい場合は、その余りの部分は、leftover の領域に割り当てられます。この leftover は、最後に表示されます。

コールバック

Item 値の変更に対応させるには、XmNxrtGaValueChangedCallback を使用してください。これは、Item の値が変更された時に呼び出されるコールバックのリストです。XmNxrtGaValueChangedCallbackStruct のポインタが各コールバックに渡されます。詳細は第 22 章を参照してください。

13.3 サンプル・コード

下記のコードは Item を作成し、値をセットしています。

```
Widget percent; /* The parent */
Widget item; /* The Item object */

item = XtVaCreateWidget(MtCreateName(name,     "item"),
          xmXrtGaItemObjectClass,              percent,
          XmNxrtGaValue,                       DVAL(value),
          XmNxrtGaLabelExtent,                 DVAL(0.5),
          COLOR_RES(XmNbackground,             color),
          NULL);
XtAddCallback(item /* Associate this item's value change with a CB */
      XmNxrtGaValueChangedCallback,
      itemCB, /* The callback function */
      (XtPointer)percent); /* The parent Percent widget */
```

13.4 リソースの一覧

Item オブジェクトの全てのリソースの一覧です。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。

XmNbackground　　Pixel　　NO_PIXEL_VALUE　　CSG

Item の背景色。背景色はデフォルトでは NO_PIXEL_VALUE で、背景色はありません。このリソースが有効な pixel 値にセットされると、イメージが指定されていても、その背景色が表示されます。イメージはオブジェクトの中央に配置され、背景色の上に表示されます。もし、背景色が表示されることなくイメージだけを表示させたい場合は、背景色をセットしないでください。

XmNxrtGaDebug　　Boolean　　False　　CSG

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Item の背景のイメージ。イメージは、オブジェクトの中央に配置されます。このリソースは、Percent ウィジェットが Linear モードの場合にのみ使用されます。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

このリソースが `True` の場合は、Item が表示されます。

XmNxrtGaLabelExtent **double** **0.5** **CSG**

Item のラベルの配置を決定します。Item が Circular Scale にセットされている場合、このリソースの値は、Scale の `XmNxrtGaRadiusSize` 値に対応します。例えば、このリソース値が 0.5 であれば、Scale の原点と円周との距離の半分の位置から、Item が配置されます。Item が横向きの Linear Scale にセットされている場合、この値が 0.2 であると、Scale の上部より、Scale の高さの 0.2 倍分離れたところから Item が配置されます。Scale の位置は、Scale の左上角のポイントが基準になります。

XmNxrtGaValue **double** ***dynamic*** **CSG**

Item の現在の値です。この値は、セットされている Scale の *min* と *max* 値の間でなければなりません。もし、値が *max* より大きいと *max* の位置に、*min* より小さいと *min* の位置に表示されてしまいます。

XmNxrtGaValueChangedCallback **XtCallbackList** **NULL** **CSG**

Item の値が変更された時の呼び出されるコールバックのリストです。`XmNxrtGaValueChangedCallbackStruct` のポインタが各コールバックに渡されます。

13.5 手続きとマクロ

XmCreateXrtGaItem()

`XrtGaItem` オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaItem (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される `XrtGaItem` オブジェクトの親、*name* は新しく作成された `XrtGaItem` オブジェクトの名前です。*arglist* は、作成時に `XrtGaItem` に渡される `XrtArgVal` リソースと値のペアのリストで、これは `NULL` とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された `XrtGaItem` オブジェクトを返しますが、オブジェクトが作成されなかった時は `NULL` が戻ります。

XmIsXrtGaItem()

```
XmIsXrtGaItem(w)
```

wがXrtGaItemオブジェクトまたは、XrtGaItemオブジェクトのサブクラスの時は、True を返すマクロです。

14

Linear Scale オブジェクト

XrtGaLinearScale ■ Linear Scale の作成
ユーザ・インタラクション ■ サンプル・コード
リソースの一覧 ■ 手続きと方法

14.1 XrtGaLinearScale

Linear Scale は、Linear Gauge や Linear Dial の中央に使用されているオブジェクトです。オブジェクトのサイズは、セットされるウィジェットの幅と高さによってコントロールすることができます。Scale には、Tick や、Range、Indicator を追加して、値をわかりやすく表示することができます。

Indicator の設定は、ダイナミックに変化するデータに対応するようコールバックを使用して操作することができます。また、Range ごとに異なる色をセットして、特定な値が目立つようにすることができます。

14.2 Linear Scale の作成

Linear Scale は、Linear Gauge や Linear Dial の中央に使用されるオブジェクトですので、Lienar Scale オブジェクトのサイズは、それがセットされるウィジェットのサイズを考慮して決定してください。テキスト等のその他のオブジェクトを配置する場合は、それらのオブジェクトの位置やサイズも考慮する必要があります。Linear Scale のサイズが決定したら、次に、Linear Scale のリソースをセットしてください。

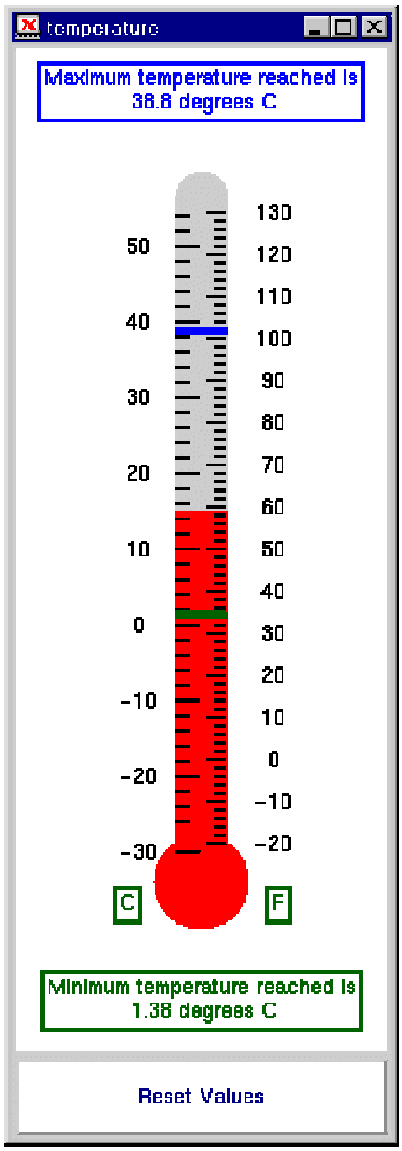


図26　摂氏と華氏を表示したLinear Gauge

14.2.1 Linear Scale のサイズと位置

Linear Scale は目に見えるオブジェクトではありませんが、サイズや位置などを指定することができます。実際に目に見える、Range、Tick、Indicator 等のオブジェクトは、Linear Scale に対応するようにその位置を調整します。

Scale の位置は、その左上角の場所を `XmNx` と `XmNy` で指定し、サイズは `XmNheight` と `XmNwidth` で指定します。

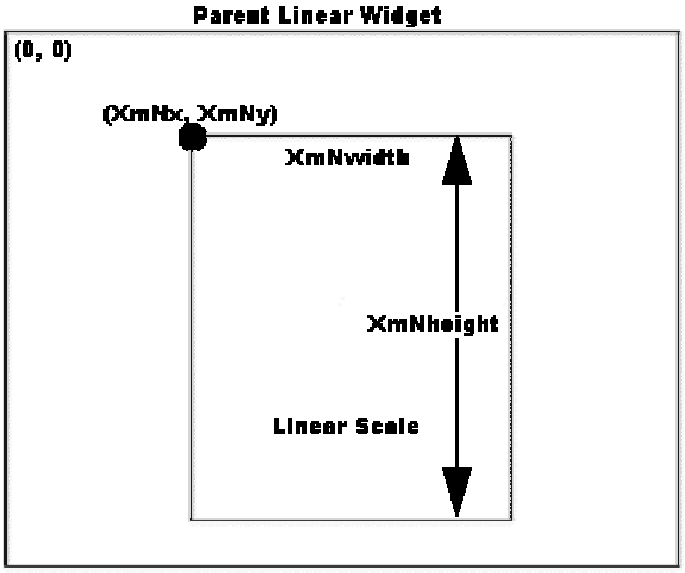


図27　Linear Scale のサイズと位置の指定

Linear Scale は、横または縦方向に配置したり、Scale の値の増加する方向を逆に指定することができます。XmNxrtGaOrientation を XmHORIZONTAL にセットした場合は値は左から右に増加し、XmVERTICAL にセットした場合は値は下から上に増加します。Scale の値が増加する方向を逆にする場合は、XmNxrtGaDirection を XRTGA_DIRECTION_BACKWARD にセットします。このようにセットされた Scale では、横方向では値は右から左に、縦方向では値は上から下に増加していきます。

14.3　ユーザ・インタラクション

XmNxrtGaAllowTransform を True にセットすることにより、エンドユーザは、Scale を移動したり、リサイズすることができるようになります。

コールバック

これらのトランスフォーメーションは、XmNxrtGaTransformCallback によってもコントロールできます。詳細は、第 22 章を参照してください。

14.4　サンプル・コード

下記のコードでは、Scale がセットされるウィジェットを指定し、Linear Scale のリソース値をセットしています。

```
Widget linear_gauge; /* The parent linear gauge Widget. /*
/*Create Basic Scale:*/
 scale = XtVaCreateWidget("scale",
     xmXrtGaLinearScaleObjectClass,   linear_gauge,
     XmNxrtGaMax,                     DVAL(60.0),
     XmNxrtGaMin,                     DVAL(-40.0),
     XmNxrtGaOrientation,             XmVERTICAL,
```

```
XmNxrtGaDirection,          XRTGA_DIRECTION_FORWARD,
XmNwidth,                   100,
XmNheight,                  450,
XmNx,                       50,
XmNy,                       75,
NULL);
```

14.5 リソースの一覧

Linear Scale オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** **NO_PIXEL_VALUE** **CSG**

Scale の背景色。このリソースがセットされていない場合、背景色は描画されません。

XmNheight **Dimension** ***dynamic*** **CSG**

Linear Scale の高さ。デフォルトは親ウィジェットの高さです。

XmNwidth **Dimension** ***dynamic*** **CSG**

Linear Scale の幅。デフォルトは親ウィジェットの幅です。

XmNx **Position** **0** **CSG**

Linear Scale の左上角の x 座標です。

XmNxrtGaAllowTransform **Boolean** **True** **CSG**

エンドユーザがスケールを移動したりスケーリングできるかどうかを指定します。キーボード・アクションは、スケーリングは*<Btn1Drag>*、移動は *Shift<Btn1Drag>*です。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDirection **XrtGaDirection** **XRTGA_DIRECTION_FORWARD** **CSG**

min から *max* への方向を指定します。有効な値は、`XRTGA_DIRECTION_FORWARD` と `XRTGA_DIRECTION_BACKWARD` です。デフォルトでは、横の Scale は左から右、縦の Scale では下から上に、値が増加します。

XmNxrtGaMax **double** **1.0** **CSG**

Scale の最大値。

XmNxrtGaMin **double** **0.0** **CSG**

Scale の最小値。

XmNxrtGaOrientation **unsigned char** **XmHORIZONTAL** **CSG**

オブジェクトの向き。有効な値は XmHORIZONTAL または XmVERTICAL です。

XmNxrtGaTransformCallback **XtCallbackList** **NULL** **CSG**

ユーザが Scale をズームまたは移動したときに呼び出されるコールバックのリストです。XmNxrtGaTransformCallbackStruct のポインタが各コールバックに渡されます。

XmNy **Position** **0** **CSG**

Linear Scale の左上角の y 座標。

14.6 手続きとマクロ

XmCreateXrtGaLinearScale()

XrtGaLinearScale オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaLinearScale (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaLinearScale オブジェクトの親、*name* は新しく作成された XrtGaLinearScale オブジェクトの名前です。*arglist* は、作成時に XrtGaLinearScale に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaLinearScale オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaLinearScale ()

```
XmIsXrtGaLinearScale (w)
```

w が XrtGaLinearScale オブジェクトまたは、XrtGaLinearScale オブジェクトのサブクラスの時は、True を返すマクロです。

15

Marker オブジェクト

XrtGaMarker ■ Marker のリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

15.1 XrtGaMarker

Marker は、Circular Dial 上にある Switch オブジェクトの現在の位置をハイライトするときに使用されます。図 28 には、Circular Dial の 270 度の位置に三角の Marker が表示されています。Marker は Scale 上の値に沿って移動します。スケールが連続モードの場合は Marker はスムーズに移動し、非連続モードの場合は指定された値の場所にのみ表示されるようになります。

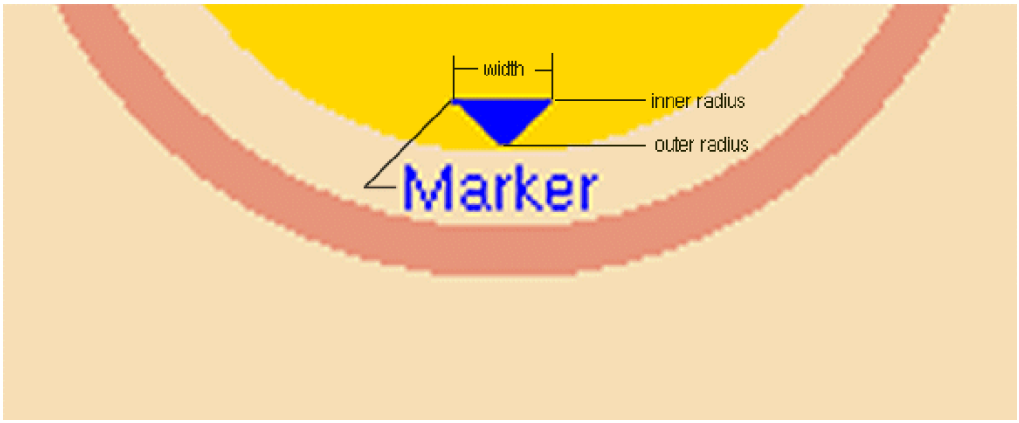


図 28　三角の Marker

Marker の inner radius(`XmNxrtGaInnerRadius`)と
outer radius(`XmNxrtGaInnerRadius`)は Scale の半径に対する割合として指定され、幅(`XmNwidth`)は、Marker が矢印または三角の場合、pixel で指定されます。inner raius が outer radius より大きい場合は、inner と outer の値は入れ替わります。

図 29 は、inner radius、outer radius、幅を使用して、Marker のサイズと位置をコントロールしている例です。

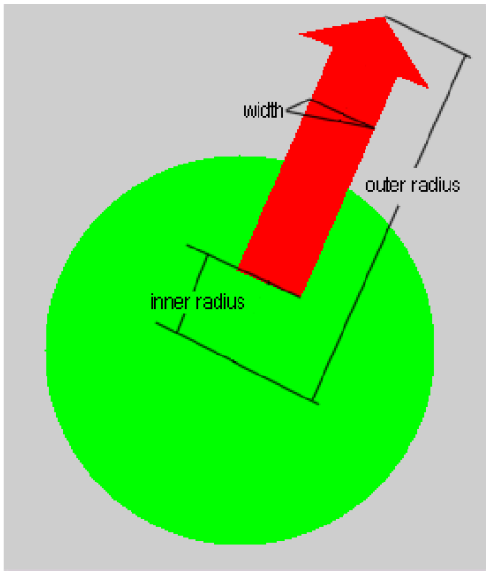


図 29　矢印の Marker

15.2　Marker のリソースについて

デバッグ情報を得るには、XmNxrtGaDebug を True にしてください。

独自の Marker を表示するには、XmNxrtGaDrawFunc に描画ファンクションに対するリファレンスを指定してください。詳細は P181 を参照してくだし。

Marker にイメージを表示するには XmNxrtGaImage を使用し、Marker の移動に従ってイメージも回転させるには、XmNxrtGaImageRotate を True にしてください。

Marker の表示、非表示は XmNxrtGaIsShowing で指定できます。

15.3　サンプル･コード

下記のコードでは、Marker オブジェクトのほとんどのリソースをセットしています。

```
/* Create triangular marker for a circular dial */
ld_marker =
 XtVaCreateWidget("ld_marker",
         xmXrtGaMarkerObjectClass,    light_dial,
         XmNxrtGaCircularScale,       ld_scale,
         XmNwidth,                    10,
         COLOR_RES(XmNbackground,     "red"),
         XmNxrtGaInnerRadius,         DVAL(1.5),
         XmNxrtGaOuterRadius,         DVAL(0.55),
         XmNxrtGaMarkerType,      XRTGA_MARKER_TYPE_TRIANGLE,
         XmNxrtGaIsShowing,       True,
         NULL);
```

15.4 リソースの一覧

Marker オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

Marker オブジェクトの背景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNwidth **Dimension** **4** **CSG**

Pixel で指定された Marker の一番広い部分の幅。Marker が円形やイメージの場合は、このリソースは無視されます。

XmNxrtGaCircularScale **XrtGaObject** **NULL** **CSG**

オブジェクトがセットされる Circular Scale。この値が `NULL` の場合は、オブジェクトは描画されません。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDrawFunc **XrtGaDrawFunc** **NULL** **CSG**

オブジェクトの描画ファンクション。この値が `NULL` でなければ、このファンクションがオブジェクトの描画に使用されます。`XmNxrtGaMarkerType` は無視されます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Marker の型がイメージの場合に表示される背景のイメージ。イメージは、Marker の inner と outer radius の間の中間に表示されます。イメージは、親の Switch オブジェクトの値によっては回転しません。

XmNxrtGaImageRotate **Boolean** **True** **CSG**

角度が変更された時に、イメージを回転させるかどうか指定します。

XmNxrtGaInnerRadius **double** **0.85** **CSG**

Marker の inner radius。この値は、それがセットされている Circular Scale の `XmNxrtGaRadiusSize` 値に対応します。例えば、このリソースの値が 0.5 であると、Marker の inner radius は、Scale の原点と円周の半分にになります。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Marker の表示、非表示を指定します。

XmNxrtGaMarkerType **XrtGaMarkerType** **XRTGA_MARKER_TYPE_CIRCLE** **CSG**

Marker の型。有効な値は、円(circle)、三角(triangle)、矢印(arrow)とイメージです。

XmNxrtGaOuterRadius **double** **0.95** **CSG**

Marker の outer radius。この値は、それがセットされている Circular Scale の XmNxrtGaRadiusSize 値に対応します。例えば、このリソースの値が 1.5 であると、Marker の outer radius は、Scale の半径のサイズより 1.5 倍分、Scale の原点から離れた場所に位置することになります。

15.5 手続きとマクロ

XmCreateXrtGaMarker()

XrtGaMarker オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaMarker (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaMarker オブジェクトの親、*name* は新しく作成された XrtGaMarker オブジェクトの名前です。*arglist* は、作成時に XrtGaMarker に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaMarker オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaMarker ()

```
XmIsXrtGaMarker (w)
```

w が XrtGaMarker オブジェクトまたは、XrtGaMarker オブジェクトのサブクラスの時は、True を返すマクロです。

16

Needle オブジェクト

XmNxrtGaNeedle ■ Needle オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

16.1 XmNxrtGaNeedle

Needle は、Circular Gauge ウィジェット内の Circular Scale オブジェクトに表示されるポインタです。ひとつの Circular Scale では、複数の Needle を表示させることができます。各 Needle は値を持ち、その値によって Scale 上のどこに表示されるかが決定されます。

Needle は、クリックまたはドラッグのアクションによっても位置を変更することができます。Circular Gauge には、マウス・アクションに対応するインタラクション・コールバックがあります。Needle の値が変更されると、value chagned コールバックが呼び出され、ユーザは変更を不許可にしたり、制限することができます。

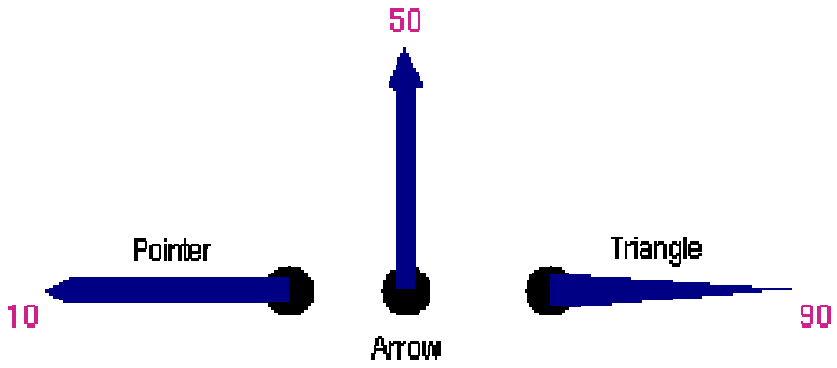


図30　三つのビルトインされた形

16.2 Needle オブジェクトのリソースについて

Needle の形

図 30 にあるように、Needle には、矢印、ポインタ、トライアングルという三つの形が用意されています。Needle の長さを指定する XmNxrtGaLength は、それがセットされる Circular Scale に対応します。Needle は、Circular Scale の中央から外方向に向かって配置されます。例えば、XmNxrtGaLength が 1.0 であれば、Needle の先は、セットされている Circular Scale の円周上に来ます。Needle の幅は XmNwidth でセットしてください。

Needle に上記以外の形を使用する場合は、独自の描画ファンクションを使用してください。詳細は P181 を参照してください。

Needle のサイズは自由に変更できますが、Needle がウィジェットの外まで伸びる長さになると、はみ出た部分は切り取られます。

Needle の色

Needle の色は XmNbackground を使用してセットできます。

Needle の表示、非表示

Needle は、XmNxrtGaIsShowing を True で表示、False で非表示にすることができます。

マウスによる Needle の配置

XmNxrtGaInteractionType を使用して、Circular Gauge 内でどのように Needle がマウス・アクションに反応するかを指定することができます。有効な値は *snap-to-mouse*、*relative-to-mouse*、*none* です。

16.3 サンプル・コード

Needle は、その作成時にリソースを全てセットできます。

```
/*Create Needle*/
interaction_type = XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE;
needle_type = XRTGA_NEEDLE_TYPE_POINTER;
needle = XtVaCreateWidget("needle",
      COLOR_RES(XmNbackground, "navyblue"),
      XmNwidth, 5,
      XmNxrtGaCircularScale,       scale,
      xmXrtGaNeedleObjectClass,    cgauge,
      XmNxrtGaInteractionType,     interaction_type,
      XmNxrtGaIsShowing,           True,
      XmNxrtGaLength,              1.0,
      XmNxrtGaNeedleType,          needle_type,
      XmNxrtGaValue,               33,
      NULL);
```

コールバック

Needle には、値が変化した時に呼び出される XmNxrtGaValueChangedCallback が

登録されています。

各コールバックには、XmNxrtGaValueChangedCallbackStruct のポインタが渡されます。詳細は第 22 章を参照してください。

16.4 リソースの一覧

Needle オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

Needle オブジェクトの背景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNwidth **Dimension** **0** **CSG**

Pixel で表された、オブジェクトの一番広い幅。この値が 0 の場合は、オブジェクトの形が何であっても、可能な限り細い線のみが表示されます。

XmNxrtGaCircularScale **XrtGaObject** **NULL** **CSG**

Needle が表示される Circular Scale。この値が NULL であると、オブジェクトは描画されません。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDrawFunc **XrtGaDrawFunc** **NULL** **CSG**

オブジェクトの描画ファンクション。値が NULL でなければ、このファンクションがオブジェクトの描画に使用され、XmNxrtGaNeedleType は無視されます。

XmNxrtGaInteractionType **XrtGaInteractionType** **XRTGA_INTERACTION_TYPE_NONE** **CSG**

オブジェクトに対するマウス・インタラクションの種類です。有効な値は、XRTGA_INTERACTION_TYPE_NONE、XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE、XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE です。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Needle の表示、非表示を指定します。

XmNxrtGaLength **double** **1.0** **CSG**

Needle の長さを指定します。この値は、それがセットされている XmNxrtGaRadiusSize に対応します。例えば、この値が 0.5 であると、Needle は、Scale の半径の半分の長さになります。Needle は常に、Scale の原点から描画されます。

XmNxrtGaNeedleType XrtGaNeedleType XRTGA_NEEDLE_TYPE_POINTER CSG

Needle の形。有効な値は、XRTGA_NEEDLE_TYPE_POINTER、XRTGA_NEEDLE_TYPE_TRIANGLE、XRTGA_NEEDLE_TYPE_ARROW です。

XmNxrtGaValue double dynamic CSG

Needle の現在値。この値は、それがセットされている Scale の *min* と *max* 値の間でなければなりません。実際に表示される値は Scale の範囲内でなければならないので、最大値より大きい値は *max* の位置に、最小値より小さい値は *min* の位置に表示されてしまいます。

XmNxrtGaValueChangedCallback XtCallbackList NULL CSG

値が変更された時に呼び出されるコールバックのリスト。XmNxrtGaValueChangedCallbackStrucnt のポインタが各コールバックに渡されます。

16.5 手続きとマクロ

XmCreateXrtGaNeedle()

XrtGaNeedle オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaNeedle (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaNeedle オブジェクトの親、*name* は新しく作成された XrtGaNeedle オブジェクトの名前です。*arglist* は、作成時に XrtGaNeedle に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaNeedle オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaNeedle ()

```
XmIsXrtGaNeedle (w)
```

w が XrtGaNeedle オブジェクトまたは、XrtGaNeedle オブジェクトのサブクラスの時は、True を返すマクロです。

17

Range オブジェクト

XrtGaRange ■ Range オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

17.1 XrtGaRange

Range オブジェクトは、Circular Gauge、Circular Dial、Linear Gauge 等に使用され、これらの Scale の全体または、部分をマークします。Range は、Circular Scale や Linear Scale にも指定することができます。

図 31 は、Circular Scale 上に表示された二つの Range オブジェクトです。薄い色のオブジェクトは全体に表示されています。濃い色のオブジェクトは、目盛りが 20 から 80 の部分の範囲に表示されています。この図に表示されているように、Range の開始値、終了値、inner extent、outer extent がオブジェクトの形を決定します。

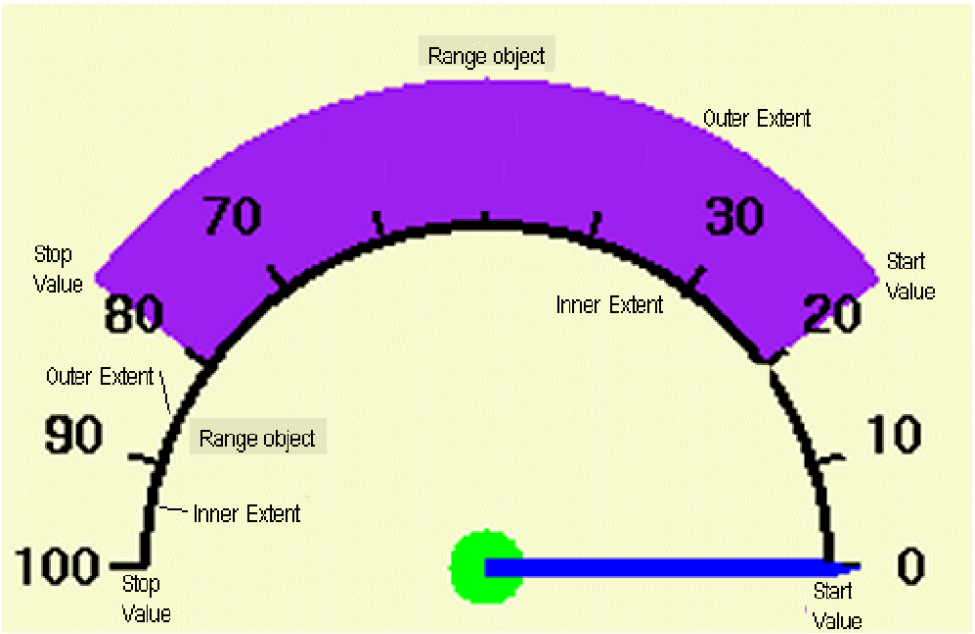


図31　Circular Scale 上の二つの Range オブジェクト

Linear Scale 上の Range オブジェクトも、Circular Scale の場合と同様です。図 32 は、Linear Gauge ウィジェットです。明るい色はウィジェットのサイズを示しています。濃い色の Range オブジェクトの inner extent は 0 にセットされ、outer extent は 1.0 にセットされていますので、このオブジェクトはウィジェットにセットされている Linear Scale のサイズを示しています。開始値と終了値にはデフォルトが使用されていますので、Range と Linear Scale のサイズは同じになっています。

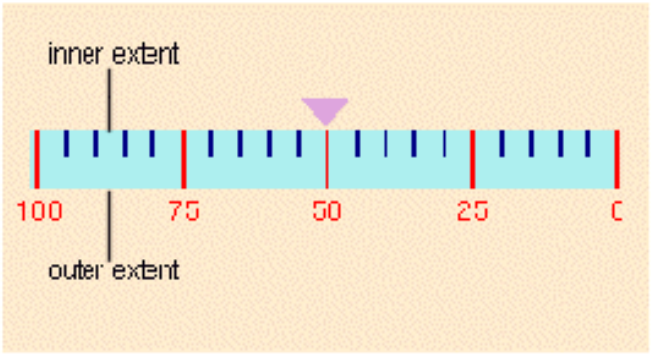


図32　Liner Scale と同じサイズの Range オブジェクト

17.2　Range オブジェクトのリソースについて

Range の色の設定には `XmNbackground` を使用してください。

Circular Scale に Range をセットする

Range オブジェクトは、`XmNxrtGaScale` を使用して、親の Scale にセットしてください。

Range は、それがセットされている Scale の領域上に表示されます。`XmNxrtGaInnerExtent` と `XmNxrtGaOuterExtent` は、Range の幅をコントロールします。これらのリソースの値は、セットされている Scale のサイズに基づいた割合です。例えば、inner extent が 0.75、outer extent が 1.25 の Range の幅は、セットされている Linear Scale の幅(向きによっては高さ)の半分、または Circular Scale の半径の半分の長さです。Range が Linear Scale にセットされている場合、長さは縦のスケールでは左側から、横のスケールでは上部から測定されます。

図 31 には、Circular Scale 上に二つの Range オブジェクトが表示されています。0 から 100 の Range は、inner radius が 1.0、outer radius が 1.0 になっていますので、Circular Scale の円周に沿って表示されています[1]。もう一つの Range の inner radius は 1.0、outer radius は 1.2 ですので、半径の 20%分だけ円周から外にまで伸びています。

[1] inner radius と outer radius が同じ値の場合、可能な限り細い線のみが表示されます。

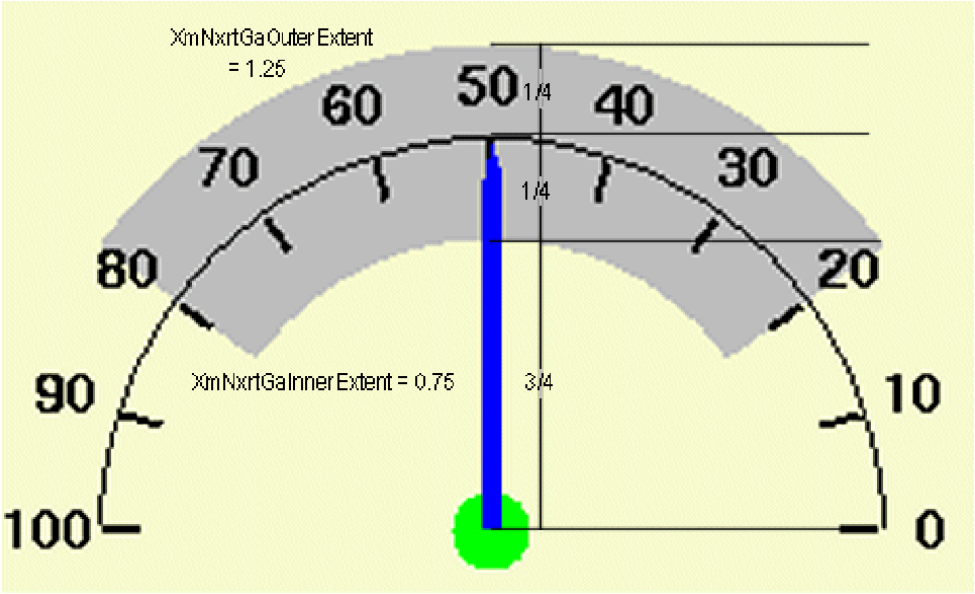


図33　Scale と Range の関係

アプリケーションが開始された時に Range の表示、非表示をコントロールするには、XmNxrtGaIsShowing を使用してください。

Range オブジェクトの形

Circular Scale では、Range は、図 33 にあるように二本の弧と二本の放射状の直線で構成されます。一方、Linear Scale では、Range には幾つかの種類の形を使用することができます。XmNxrtGaShape を使用して、矩形(デフォルト)、円形、または温度計やタンクの形を選択することができます。また、XmNxrtGaImage と XmNxrtGaImageLayout によってイメージを指定することもできます。

XmNxrtGaStartValue、XmNxrtGaStopValue、XmNxrtGaOffset は、Range オブジェクトにマークされる Scale の値をセットします。

Scale の範囲外への表示

Circular Gauge ウィジェットにセットされている Circular Scale の領域の外へ、Range をシフトさせることができます。

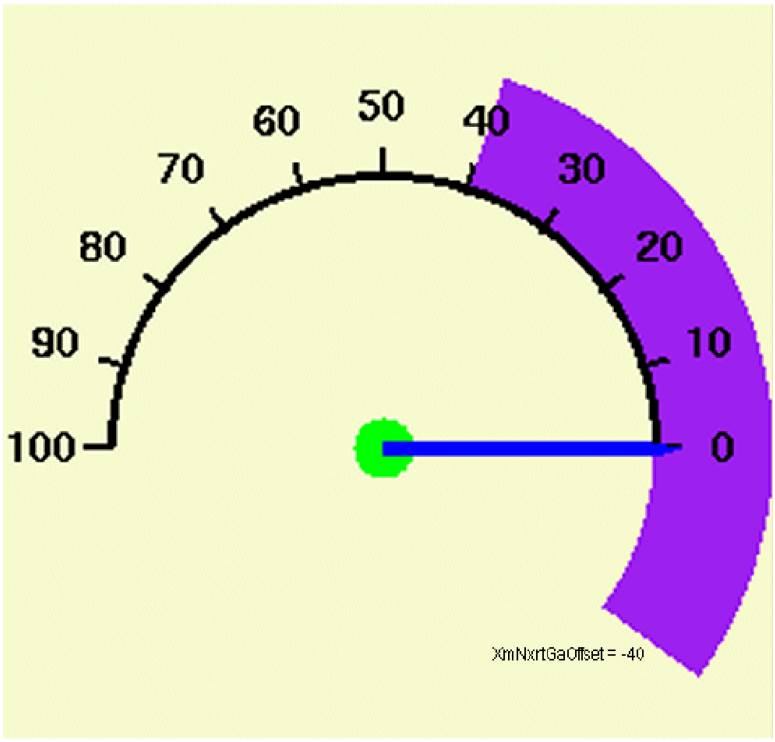


図34　オフセットを調節してScale の外まで表示した(図31 と同じ)Range オブジェクト

図 34 のように、Range をシフトするには、XmNxrtGaOffset を使用してください。この図では、オフセットに負の数を使用することにより、Range が Scale の開始点より前から表示されるようになっています。このオフセットは、スケールの範囲を最大として、左右どちらの方向にも設定することができます。

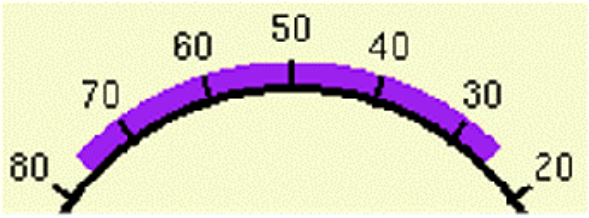


図35　二つのRange オブジェクト

17.3　サンプル･コード

下記のコードを参考にして、Range オブジェクトのリソースをセットしてください。最初の例では、親が Linear Scale にセットされています。

```
Widget lineardial; /* The parent widget for the Range object */
Widget ld_scale; /* The parent scale for the Range object */

/* Create Range */
range =
```

```
XtVaCreateWidget("range",
      XmNbackground,            "green"
      xmXrtGaRangeObjectClass, lineardial,
      XmNxrtGaScale,            ld_scale,
      XmNxrtGaInnerExtent,      DVAL(0.9),
      XmNxrtGaOuterExtent,      DVAL(1.0),
      XtVaTypedArg,             XmNxrtGaImage,
      XmRString,       "$XRTHOME/src/icons/shading.xpm",
      strlen("$XRTHOME/src/icons/shading.xpm"+1),
      XmNxrtGaImageLayout,      XRTGA_IMAGE_LAYOUT_TILED,
      XmNxrtGaIsShowing,        True,
      XmNxrtGaOffset,           DVAL(0.0),
      XrtGaShape,               XRTGA_SHAPE_RECTANGLE,
      XmNxrtGaStartValue,       DVAL(0.0),
      XmNxrtGaStopValue,        DVAL(10.0),
      NULL);
```

下の例では、親が Circular Scale にセットされています。

```
Widget circulargauge; /* The parent widget for the Range object */
Widget cd_scale; /* The parent scale for the Range object */

/* Create Range */
range =
 XtVaCreateWidget("range",
      XmNbackground,            "blue"
      xmXrtGaRangeObjectClass, circulargauge,
      XmNxrtGaScale,            cd_scale,
      XmNxrtGaInnerExtent,      DVAL(0.9),
      XmNxrtGaOuterExtent,      DVAL(1.0),
      XmNxrtGaIsShowing,        True,
      XmNxrtGaOffset,           DVAL(0.0),
      XmNxrtGaStartValue,       DVAL(0.0),
      XmNxrtGaStopValue,        DVAL(10.0),
      NULL);
```

17.4 リソースの一覧

Range オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

Range オブジェクトの背景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Range の型がイメージで、セットされている Scale が Linear の時に使用されるイメージです。イメージは、Range が定義されている領域の中央に配置されます。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_TILED** **CSG**

背景のイメージの配置。有効な値は、`XRTGA_IMAGE_LAYOUT_CENTERED` と `XRTGA_IMAGE_LAYOUT_TILED` です。

XmNxrtGaInnerExtent **double** **1.0** **CSG**

Range の inner extent。セットされている Scale が Circular の場合、この値はその Scale の `XmNxrtGaRadiusSize` 値に対応します。例えば、Circular Scale でこの値が 0.5 の場合、inner extent は Scale の原点と半径の半分の距離から開始されます。横向きの Linear Scale では、この値が 0.2 であると、inner extent は、Scale の高さの 0.2 倍分だけ、Scale の上部から離れたところに位置します。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Range オブジェクトの表示、非表示を指定します。

XmNxrtGaOffset **double** **0.0** **CSG**

Circular Range では、オブジェクトの表示が始まる位置を指定します。このリソースを使用して Range をシフトし、Scale の最小値または最大値の外に表示させることができます。オフセットは、Scale の範囲(Scale の *min* 値と *max* 値との差)を最大として、左右どちらの方向にもセットすることができます。

XmNxrtGaOuterExtent **double** **1.0** **CSG**

Range の outer extent。セットされている Scale が Circular の場合、この値はその Scale の `XmNxrtGaRadiusSize` 値に対応します。例えば、Circular Scale でこの値が 1.5 は、Scale の半径の 1.5 倍を意味します。横向きの Linear Scale では、値が 0.8 であると、outer extent は、Scale の高さの 0.8 倍分だけ、Scale の上部から離れた場所に来ます。

XmNxrtGaScale **XrtGaObject** **NULL** **CSG**

Range オブジェクトにセットされる Circular または Linear Scale。この値が `NULL` の場合は、対応する Range は描画されません。

XmNxrtGaShape **XrtGaShape** **XRTGA_SHAPE_RECTANGLE** **CSG**

Range の形。このリソースは、セットされている Scale が Circular の場合は無視されます。有効な値は、`XRTGA_SHAPE_RECTANGLE`、`XRTGA_SHAPE_CIRCLE`、`XRTGA_SHAPE_THERMOMETER`、`XRTGA_SHAPE_TANK`、`XRTGA_SHAPE_IMAGE` です。

XmNxrtGaStartValue **double** ***dynamic*** **CSG**

Range の開始値。この値は、セットされている Scale の *min* と *max* の間でなければなりません。表示される値は Scale の範囲内に限られますので、最小値より小さい値は *min* に、

最大値より大きい値は max に表示されてしまいます。このリソースのデフォルト値は、Scale の最小値にセットされています。

XmNxrtGaStopValue　　double　　*dynamic*　　CSG

Range の終了値。この値は、セットされている Scale の *min* と *max* の間でなければなりません。表示される値は Scale の範囲内に限られますので、最小値より小さい値は *min* に、最大値より大きい値は max に表示されてしまいます。このリソースのデフォルト値は、スケールの最大値にセットされています。

17.5　手続きとマクロ

XmCreateXrtGaRange()

XrtGaRange オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaRange (
     Widget       parent,
     char *       name,
     ArgList      arglist,
     Cardinal     argcount
     )
```

parent は作成される XrtGaRange オブジェクトの親、*name* は新しく作成された XrtGaRange オブジェクトの名前です。*arglist* は、作成時に XrtGaRange に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaRange オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaRange ()

```
XmIsXrtGaRange (w)
```

w が XrtGaRange オブジェクトまたは、XrtGaRange オブジェクトのサブクラスの時は、True を返すマクロです。

18

State オブジェクト

XrtGaState ■ State オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

18.1 XrtGaState

State は、Circular Dial ウィジェット内の Switch オブジェクトにセットされるオブジェクトです。Switch が非連続モードの場合、Switch を合わせることができる値を *state* と呼びます。

State オブジェクトの親は常に `XrtGaCircularDial` ウィジェットです。また、State が使用される Circular Scale は必ずセットされている必要があります。非連続モードの Dial では、State オブジェクトのリストが、設定可能な状態(State)を定義します。新しく作成された State は、親の State リストに追加されます。このリストは、ウィジェットの `XmNxrtGaStateList` リソースに記憶されます。このリストは read-only でユーザは内容を変更することはできません。

State のラベルは `XmNxrtGaText` で指定します。このラベル内でのテキストの配置は、Switch から継承されます。`XmNxrtGaAlignment` によって、このデフォルトの方法を State ごとに設定することができます。詳細は、この章の最後の `XmNxrtGaAlignment` の章を参照してください。

デフォルトでは、Scale の開始から終了の角度の間に均等に State が配置されます。この配置は、`XmNxrtGaAngle` によって State ごとに変更することができます。

18.2 State オブジェクトのリソースについて

State のラベルに使用されるフォントは、`XmNfontList` で指定されます。

Scale への配置

デフォルトでは、State は、Circular Scale の開始と終了の角度の間に均等に配置されます。`XmNxrtGaAngleUseDefault` が `True` であれば、ある State における角度は自動的に計算されます。

XmNxrtGaAngle には、異なる角度を指定することができます。その場合は、副作用で XmNxrtGaAngleUseDefault は False にセットされます。XmNxrtGaAngleUseDefault を True にリセットすれば、ウィジェットは、角度を計算するようになります。

ラベル内のテキストの行揃え

State には、XmNxrtGaText でラベルを貼り付けることができます。ストリングは、\n などのフォーマット・シンボルを含むことができ、XmNxrtGaAlignment を使用して行揃えをコントロールすることができます。行揃えの三つの例は、図 36 を参照してください。*state1* は左揃え(XRTGA_ALIGNMENT_BEGINNING)、*state2* と *state3* は中央揃え(XRTGA_ALIGNMENT_CENTER)、*state4* と *state5* は右揃え(XRTGA_ALIGNMENT_END)になっています。

線の表示

State は、Scale 上での位置が指定されており、この部分からラベルに対して放射上に伸びる線を引くことができます。線の開始点は XmNxrtGaLineInnerRadius で指定され、XmNxrtGaLineOuterRadius で指定された値で終了します。この点から、XmNxrtGaOffsetX と XmNxrtGaOffsetY をセットしてさらに線を追加することができます。図 36 の State5 を参照してください。State5 の角度は 180 度なので、線は水平方向に表示されています。この線の outer radius は 1.2 なので、Scale の半径より 20%分伸びています。この点から、y-offset を 90pixel 追加し、x-offset は 40pixel 減少させます。Y 値は下方向に、X 値は右方向に増加しますので、このオフセットの設定により、線は下方向 90pixel、左方向 40pixel の方向に向かって伸びています。

State オブジェクトの配置

デフォルトでは、inner radius、outer radius、offset は、XmNxrtGaLineInnerRadius、XmNxrtGaLineOuterRadius、XmNxrtGaOffsetType、XmNxrtGaOffsetDistance によってコントロールされています。

XmNxrtGaLineInnerRadiusUseDefault または XmNxrtGaLineOuterRadiusUseDefault が True の場合、State は、inner または outer radius の値を Circular Dial の XmNxrtGaLineInnerRadius と XmNxrtGaLineOuterRadius から取得します。State の XmNxrtGaOffsetUseDefault が True の場合は、x-offset と y-offset の値は Circular Dial の XmNxrtGaOffsetType によって決定されます。オフセットの種類が XRTGA_OFFSET_TYPE_CIRCULAR であれば、x-offset と y-offset の値は 0 にセットされます。XRTGA_OFFSET_TYPE_HORIZONTAL または XRTGA_OFFSET_TYPE_VERTICAL であれば、XmNxrtGaOffsetDistance が、Circular Scale の中央を通る横または縦の線から同じ距離の位置に State を配置します。この例は P26 の図 6 を参照してください。オフセットの種類が XRTGA_OFFSET_TYPE_AUTOMATIC であれば、x-offset と y-offset には、最も適した値が選択されます。

XmNxrtGaAlignmentUseDefault が True であれば、オフセットの種類に基づいて適したテキストの配置が行われます。inner radius、outer radius、x-offset、y-offset の値のどれかがセットされれば、対応する UseDefault リソースは False にセットされます。

線の種類はXmNxrtGaLinePatternでセットします。図36には、選択できる線の種類が幾つか例として表示されています。この線の表示、非表示はXmNxrtGaLineDisplayでコントロールしてください。

色

テキスト・ラベルの(ハイライトされていない状態の)色をセットするには、XmNforegroundを使用してください。

表示と非表示

Stateの表示、非表示は、XmNxrtGaIsShowingでコントロールできます。

18.3 サンプル・コード

下記のサンプル・コードでは、Stateオブジェクトのほとんどのリソースがセットされています。図36では、それぞれ異なる線の種類、inner radius、オフセット等をもった五つのStateが表示されています。*switch1*から、配置は、*beginning*、*center*、*center*、*end*、*end*となっています。

```
Widget circular_dial; /* The parent circular dial */
Widget cd_scale; /* The state is attached to this scale */
Position xpos1 = 10, ypos1 = 10;
 /* Create States */
 state1 = XtVaCreateWidget("state1",
     xmXrtGaStateObjectClass,     circular_dial,
     XmNxrtGaCircularScale,       cd_scale,
     XmNxrtGaAlignment,           XRTGA_ALIGNMENT_END,
     XmNxrtGaAlignmentUseDefault, False,
     XmNxrtGaAngle,               DVAL(0.0),
     XmNxrtGaAngleUseDefault,     False,
     XmNxrtGaLineDisplay,         True,
     XmNxrtGaLinePattern,         XRTGA_LPAT_DASH_DOT,
     XmNxrtGaLineThickness,       2,
     XmNxrtGaLineInnerRadiusUseDefault,   False,
     XmNxrtGaLineInnerRadius,             DVAL(0.4),
     XmNxrtGaLineOuterRadiusUseDefault,   False,
     XmNxrtGaLineOuterRadius,             DVAL(1.2),
     XmNxrtGaOffsetX,                     xpos1,
     XmNxrtGaOffsetY,                     ypos1,
     XmNxrtGaOffsetUseDefault,            False,
     XmNxrtGaText,                        "state 1",
     COLOR_RES(XmNforeground,             "blue"),
     XmNxrtGaIsShowing,                   True,
     NULL);
```

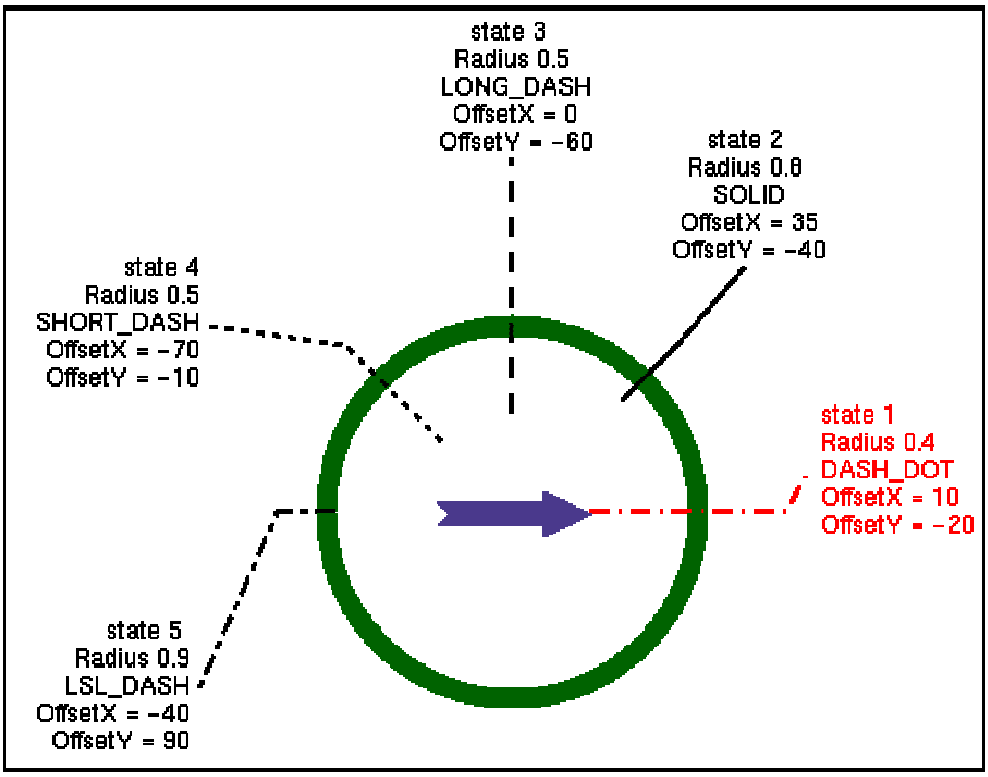


図36　五つの State を持った Switch

18.4　リソースの一覧

State オブジェクトの全てのリソースの一覧です。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。

XmNfontList　　XmFontList　　"fixed"　　CSG

State に表示されるテキストのフォント。

XmNforeground　　Pixel　　*inherited*　　CSG

State オブジェクトの前景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。この色は、オブジェクトのテキストや線に使用されます。

XmNxrtGaAlignment　　XrtGaAlignment　　*dynamic*　　CSG

State を説明するテキストの行揃えを定義します。対応する `UseDefault` が `True` であると、Switch の左側の State に対しては右揃え、Switch の右側の State に対しては左揃え、Switch の上または下にある State に対しては中央揃えにセットされます。このリソースをセットすると、対応する `UseDefault` リソースは `False` にセットされます。

XmNxrtGaAlignmentUseDefault　　Boolean　　True　　CSG

`True` の場合、`XmNxrtGaAlignment` はデフォルト値を使用します。`XmNxrtGaAlignment` に何もセットされていない場合(デフォルトが計算される前)は、このリソースは `False` にできません。

XmNxrtGaAngle **double** ***dynamic**** **CSG**

このリソースは、State が描画される角度を定義します。デフォルトでは、この角度は、セットされる Circular Scale の開始と終了の角度、リスト上の State の数と対象となる State のそのリストでの位置から計算されます。このリソースをセットすると、対応する `UseDefault` リソースは `False` にセットされます。

*`XmNxrtGaAngle` は、セットされている Circular Scale オブジェクトからデフォルト値を取得します。

XmNxrtGaAngleUseDefault **Boolean** **True** **CSG**

`True` の場合、`XmNxrtGaAngle` リソースはデフォルト値を使用します。`XmNxrtGaAngle` に何もセットされていない場合(デフォルトが計算される前)は、このリソースは `False` にセットできません。

XmNxrtGaCircularScale **XrtGaObject** **NULL** **CSG**

セットされる Circular Scale。この値が `NULL` の時は、オブジェクトは描画されません。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを `XtAppWarningMessage` にプリントするかどうか指定します。`False` の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

State オブジェクトを表示するかどうか指定します。

XmNxrtGaLineDisplay **Boolean** **True** **CSG**

`True` の場合は、State に対する線が表示されます。

XmNxrtGaLineInnerRadius **double** ***dynamic **** **CSG**

State の inner radius オフセットを定義します。このリソースの値は、親の Scale オブジェクトの半径に関連しています。例えば、この値が 0.5 であると、inner radius は、Scale の半径の半分になります。`XmNxrtGaLineInnerRadiusUseDefault` が `True` であると、`XrtGaCircularDial` の `XmNxrtGaLineInnerRadius` がこの値を決定します。このリソースをセットすると副作用で、`XmNxrtGaLineInnerRadiusUseDefault` が `False` にセットされます。

*`XmNxrtGaLineInnerRadius` は、親の Circular Dial からデフォルト値を取得します。

XmNxrtGaLineInnerRadiusUseDefault **Boolean** **True** **CSG**

`True` の場合は、`XmNxrtGaLineInnerRadius` は、Circular Dial の `XmNxrtGaLineInnerRadius` から取得されたデフォルト値を使用します。`XmNxrtGaLineInnerRadius` に何もセットされていない場合(デフォルトが計算される前)は、このリソースは `False` にセットできません。

XmNxrtGaLineOuterRadius **double** ***dynamic **** **CSG**

State の outer radius オフセットを定義します。このリソースの値は、親の Scale オブジェクトの半径に関連します。例えば、この値が 1.5 であると、outer radius は、Scale の半径の 1.5 倍になります。`XmNxrtGaLineOuterRadiusUseDefault` が `True` であると、Circular Dial の `XmNxrtGaLineOuterRadius` リソースがこの値を決定します。

このリソースをセットすると、副作用で XmNxrtGaLineOuterRadiusUseDefault が False になります。

*XmNxrtGaLineOuterRadius は、Circular Dial からデフォルト値を取得します。

XmNxrtGaLineOuterRadiusUseDefault Boolean True CSG

True の場合、XmNxrtGaLineOuterRadius は Circular Dial の XmNxrtGaLineOuterRadius から取得したデフォルト値を使用します。XmNxrtGaLineOuterRadius に何もセットされていない(outer radius の値が計算されていない)場合は、このリソースは False にセットできません。

XmNxrtGaLinePattern XrtGaLinePattern XRTGA_LPAT_SOLID CSG

State に表示される線のスタイル。有効な値は、次の通りです。

XRTGA_LPAT_NONE	線は表示されない
XRTGA_LPAT_SOLID	実線
XRTGA_LPAT_LONG_DASH	長いダッシュ
XRTGA_LPAT_DOTTED	点線
XRTGA_LPAT_SHORT_DASH	短いダッシュ
XRTGA_LPAT_LSL_DASH	長いダッシュと短いダッシュ
XRTGA_LPAT_DASH_DOT	ダッシュと点線

XmNxrtGaLineThickness int 1 CSG

線幅を指定します。この値が 0 であると、可能な限り細い線が表示されます。

XmNxrtGaOffsetUseDefault Boolean True CSG

XmNxrtGaOffsetX と XmNxrtGaOffsetY にデフォルト値を使用するかどうかコントロールします。どちらかのリソースがユーザによって編集されると、XmNxrtGaOffsetUseDefault が False にセットされ、両方のオフセットはウィジェットによって計算されなくなります。XmNxrtGaOffsetX と XmNxrtGaOffsetY の両方の値がセットされないと、このリソースは False にセットすることはできません。

XmNxrtGaOffsetX Position *dynamic* CSG

Outer radius オフセットが適用された後に、このリソースは x 方向のオフセットを追加します。デフォルトでは、Circular Dial の XmNxrtGaOffsetType と XmNxrtGaOffsetDistance がこの値の決定に使用されます。このリソースをセットすると副作用で XmNxrtGaOffsetUseDefault は False にセットされます。

XmNxrtGaOffsetY **Position** ***dynamic*** **CSG**

Outer radius オフセットが適用された後に、このリソースは y 方向のオフセットを追加します。デフォルトでは、Circular Dial の XmNxrtGaOffsetType と XmNxrtGaOffsetDistance がこの値の決定に使用されます。このリソースをセットすると副作用で XmNxrtGaOffsetUseDefault は False にセットされます。

XmNxrtGaText **String** **NULL** **CSG**

State に対するストリングです。

18.5 手続きとマクロ

XmCreateXrtGaState()

XrtGaState オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaState (
     Widget       parent,
     char *       name,
     ArgList      arglist,
     Cardinal     argcount
     )
```

parent は作成される XrtGaState オブジェクトの親、*name* は新しく作成された XrtGaState オブジェクトの名前です。*arglist* は、作成時に XrtGaState に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaState オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaState()

```
XmIsXrtGaState (w)
```

w が XrtGaState オブジェクトまたは、XrtGaState オブジェクトのサブクラスの時は、True を返すマクロです。

19

Switch オブジェクト

XrtGaSwitch ■ Switch について
サンプル・コード ■ コールバック
リソースの一覧 ■ 手続きと方法

19.1 XrtGaSwitch

Switch オブジェクトは Linear Dial ウィジェットに使用されます。Linear Dial 一つにつき一つの Switch オブジェクトしか使用できず、必ず、Dial の Circular Scale にセットされている必要があります。

Switch オブジェクトには、四つの種類が用意されていますが、描画ファンクションを提供することにより、Switch を独自にデザインすることもできます。

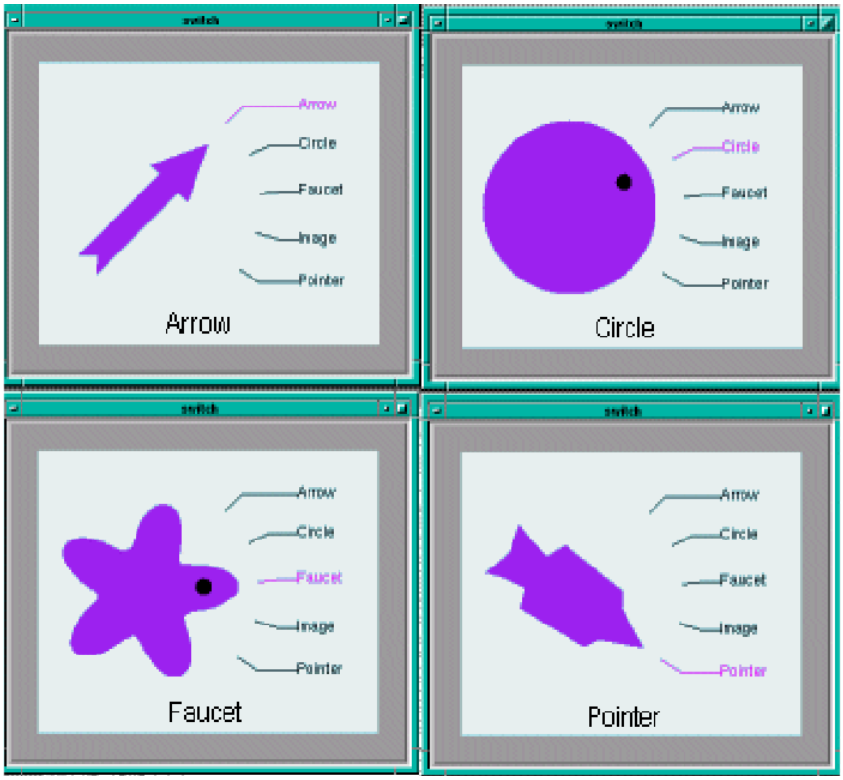


図37　用意されている四つの Switch の形状

19.2 Switch について

Circular Dial の XmNxrtGaSwitch が NULL であると、Switch 及び Switch に付属する Marker や State などは表示されません。デフォルトでは、Switch オブジェクトは作成されません。Switch が作成され、XmNxrtGaSwitch がデフォルト値の NULL であると、このリソースは自動的に新しいオブジェクトを指すようになり、ウィジェットにはこのオブジェクトが使用されるようになります。このリソースが指している Switch が破棄されれば、このリソースは NULL にセットされます。

Switch の値

Switch には連続と非連続の二つのモードがあります。XmNxrtGaContinuous を True にして設定される連続モードでは、Circular Scale の min と max の間であれば、どこにでも Switch を合わせることができます。

連続モードでも、XmNxrtGaIncrementValue を使用してインクレメントを決めることができます。例えば、Scale の *min* が 0、*max* が 100 で、インクレメントが 10 であれば、Switch は、0、10、20、…100 の値にしか合わせることはできません。これらのインクレメントに当てはまらない値がオブジェクトに与えられると、その値に一番近いインクレメントが選択されます。インクレメントが 0.0 以下であると、インクレメントは適用されません。

XmNxrtGaContinuous を False にすると、0、1、2…n-1(*n* は State の数)しか選択できません。その他の値にセットしようとしても、その値から一番近い値にジャンプします。

Multi-turn

Switch には、multi-turn(複数の回転)をセットすることができます。Switch は、XmNxrtGaContinuous を True にして必ず連続モードにし、XmNxrtGaMultipleTurnEnabled を True にして multi-turn をセットしてください。XmNxrtGaMultipleTurnValue の値は必ず 1 より大きくし、セットされる Circular Scale の開始角度を終了角度と同じにして一回転できるようにしてください。multi-turn にすると、Scale の *min* から *max* に向かって XmNxrtGaMultipleTurnValue でセットされた数分、回転します。

イメージの使用

Switch には用意された形のほかに、イメージを使用することができます。Switch に合わせてイメージも回転させる場合は、XmNxrtGaImageRotate を True にしてください。

次の例は、Switch に背景のイメージをセットしています。

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

void
set_image(Widget switch, char *file_name)
{
      XrtImage 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(image, XtRString,
                &from, XtRXrtImage, &to)) {
            fprintf(stderr,"Conversion of %s to XtRXrtImage failed\n",
                file_name);
            image = NULL;
        }
        XtVaSetValues(switch,
            XmNxrtGaImage, 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(converter)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。

19.3 サンプル・コード

下記のサンプル・コードは、Switch オブジェクトのほとんどのリソースをセットしています。

```
Widget circulardial; /* The parent widget - a circular dial*/
Widget cd_scale; /* The parent circular scale*/
XrtGaInteractionType interaction_type;

interaction_type = XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE;

/* Create Switch */
 interaction_type = XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE;
 main_switch =
     XtVaCreateWidget("main_switch",
         xmXrtGaSwitchObjectClass,    circulardial,
         XmNxrtGaCircularScale,       cd_scale,
         XmNxrtGaIsShowing,           True,
         XmNxrtGaRadius,              DVAL(0.85),
         XmNxrtGaContinuous,          True,
         XmNxrtGaIncrementValue,      DVAL(10.0),
         XmNxrtGaMultipleTurnEnabled, True,
         XmNxrtGaMultipleTurnValue,   DVAL(2.0),
         XmNxrtGaSwitchType,      XRTGA_SWITCH_TYPE_POINTER,
         XmNxrtGaInteractionType,     interaction_type,
         XmNxrtGaImageRotate,         True,
         XmNxrtGaValue,               DVAL(0.0),
         COLOR_RES(XmNbackground,     "darkslateblue"),
         NULL);
 XtAddCallback(main_switch, XmNxrtGaValueChangedCallback,
     cd_ValueCB, NULL);
```

背景のイメージは、他の fallback リソースと一緒にセットすることができます。

```
static String fallbacks[] = {
  "*.cdial.background:          grey80",
  "*.xrtGaDebug:                True",
  "*.XmXrtGaSwitch.xrtGaImage: $XRTHOME/src/icons/arrow3.xpm",
  NULL
};
```

その他の方法として、イメージをセットするファンクションを定義した後に XtVaSetValues を使用してイメージをセットすることもできます。この方法は、前の章で解説されていますので参照してください。

19.4 コールバック

XmNxrtGaValueChangedCallback コールバックは、Switch オブジェクトから登録できます。これは、値が変更された時に呼び出されるコールバックのリストです。XmNxrtGaValueChangedCallbackStruct のポインタが各コールバックに渡されます。詳細は第 22 章を参照してください。

19.5 リソースの一覧

Switch オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

Switch オブジェクトの背景の色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNxrtGaCircularScale **XrtGaObject** **NULL** **CSG**

Switch オブジェクトがセットされる Circular Scale。この値が NULL の場合は、Switch オブジェクトは描画されません。

XmNxrtGaContinuous **Boolean** **True** **CSG**

True の場合、Switch はスケールの最小値と最大値の間であればどの値でも取ることができます。False の場合は、リスト上の値から次の値にジャンプし、その他の値を取ることはできません。この場合の値のリストは 0、1、2、…n-1 で、n は State の数を指します。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDrawFunc **XrtGaDrawFunc** **NULL** **CSG**

オブジェクトを描画するファンクション。この値が NULL でなければ、このファンクションがオブジェクトの描画に使用され、XmNxrtGaSwitchType リソースは無視されます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Switch の背景のイメージ。Switch の型がイメージでこのリソースが NULL でなければ、イメージは、セットされた Circular Scale の原点に対して中央に表示されます。XmNxrtGaImageRotate は、Switch の値が変更された時にイメージも回転させるかどうか指定します。Switch の値を追う場合は、代わりにMarker を使用してください。

XmNxrtGaImageRotate **Boolean** **True** **CSG**

角度の変更に応じてイメージも回転させるかどうか指定します。

XmNxrtGaIncrementValue **double** **0.0** **CSG**

Switch が連続モードの時にインクレメントを指定します。例えば、Scale の min が 0、max が 100 で、インクレメントが 10 であれば、Switch は、0、10、20、…100 の値にしか合わせることはできません。これらのインクレメントに当てはまらない値がオブジェクトに与えられると、その値に一番近いインクレメントが選択されます。インクレメントが 0.0 以下であると、インクレメントは適用されません。

XmNxrtGaInteractionType **XrtGaInteractionType** **XRTGA_INTERACTION_TYPE_NONE** **CSG**

オブジェクトに対するマウス・インタラクションです。有効な値は、XRTGA_INTERACTION_TYPE_NONE、XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE、XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE です。

XRTGA_INTERACTION_TYPE_NONE	クリックやドラッグは何もしない
XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE	親ウィジェット内でドラッグが行われると、Switch はマウスの位置に従う
XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE	親ウィジェット内でクリックが行われると、Switch は、マウスの位置に従う

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Switch オブジェクトの表示、非表示を指定します。

XmNxrtGaMultipleTurnEnabled **Boolean** **False** **CSG**

min から max に向かって、Switch が複数回、回転することを許可します。このリソースは、XmNxrtGaContinuous が True で、その他の条件も合っている場合にのみ適用されます。詳細は、XmNxrtGaMultipleTurnValue を参照してください。

XmNxrtGaMultipleTurnValue **double** **1.0** **CSV**

このリソースを使用するには、次の条件が満たされている必要があります。

1. Switch は連続モード(XmNxrtGaContinuous は True)。
2. multi-turn が許可されている(XmNxrtGaMultipleTurnEnabled は True)。
3. セットされる Circular Scale の開始と終了の角度が同じ(一回転できる)。

4. このリソースの値を 1.0 より大きい値にセットする。例えば、2.0 がセットされると、Switch は二回転することができる。

Circular Dial でセットされている Range や Tick は、複数の回転に対応するようには表示されません。

XmNxrtGaRadius **double** **0.75** **CSG**

Switch オブジェクトの outer extent。この値は、セットされている Circular Scale の XmNxrtGaRadiusSize 値に関連しています。例えば、この値が 0.5 であると、Switch の半径は、スケールの半径の半分のサイズになります。

XmNxrtGaSwitchType **XrtGaSwitchType** **XRTGA_SWITCH_TYPE_CIRCLE** **CSG**

Switch の種類。有効な値は、XRTGA_SWITCH_TYPE_CIRCLE、XRTGA_SWITCH_TYPE_POINTER、XRTGA_SWITCH_TYPE_ARROW、XRTGA_SWITCH_TYPE_FAUCET、XRTGA_SWITCH_TYPE_IMAGE です。

独自の描画ファンクションを使用する場合は、P181 を参照してください。

XmNxrtGaValue **double** ***dynamic*** **CSG**

Switch の現在の値。Switch が連続モードの場合は、この値は、セットされているスケールの *min* と *max* の間でなければなりません。*max* より大きい値は *max* の位置に、*min* より小さい値は *min* の位置に表示されてしまいます。

Switch が非連続モードの場合は、XmNxrtGaValue は 0 と n-1(n は、親ウィジェットの State リストにある State の数)の間の整数です。整数でない値は、その近くの整数に四捨五入され、範囲の外の整数は無視されます。

XmNxrtGaValueChangedCallback **XtCallbackList** **NULL** **CSG**

値が変更された時に呼び出されるコールバックのリストです。XmNxrtGaValueChangedCallbackStruct のポインタが各コールバックに渡されます。

19.6 手続きと方法

XmCreateXrtGaSwitch()

XrtGaSwitch オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaSwitch (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaSwitch オブジェクトの親、*name* は新しく作成された XrtGaSwitch オブジェクトの名前です。*arglist* は、作成時に XrtGaSwitch に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaSwitch オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaSwitch()

```
XmIsXrtGaSwitch (w)
```

w が XrtGaSwitch オブジェクトまたは、XrtGaSwitch オブジェクトのサブクラスの時は、True を返すマクロです。

20

Text オブジェクト

XrtGaText ■ Text オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

20.1 XrtGaText

Text オブジェクトは、Circular Dial、Circular Gauge、Linear Dial、Linear Gauge、Percent ウィジェットなどに使用することができる一般的なテキスト・ボックスです。通常、Text オブジェクトは、ウィジェット内のどの場所にいくつでも貼り付けることができます。

20.2 Text オブジェクトのリソースについて

ウィジェット上での貼り付け位置は、XmNx と XmNy で指定されるテキスト・ボックスの左上角の位置で設定します。このテキスト・ボックスのサイズは XmNwidth と XmNheight で決定されます。これらのリソースの値は read-only です。

XmNbackground は Text オブジェクトの背景の色、XmNxrtGaBorder は枠線の種類、XmNxrtGaBorderWidth はその枠線の幅を指定します。

Text オブジェクトのテキスト自身は XmNforeground で指定される色で表示され、フォントは XmNfontList で指定されます。

Text オブジェクトの背景には、XmNxrtGaImage を使用してイメージを表示させることができます。詳細は、P186 を参照してください。テキスト・ボックス内でイメージを中央に配置する場合は、XmNxrtGaImageLayout を True にセットしてください。XmNxrtGaImageLayout を False にセットするとイメージはタイルのように並べられます。テキスト・ボックスがイメージに対して十分な大きさになるように、XmNxrtGaImageMinimumSize を True にセットしてください。

Text オブジェクトの表示、非表示は XmNrtGaIsShowing でコントロールします。

ストリングの種類の選択

Text には、通常の C ストリングか XmString を使用することができます。C ストリングを使用する場合は、XmNxrtGaStringMode を XRTGA_STRING_MODE_STRING にセットし、XmString を使用する場合は、XmNxrtGaStringMode を XRTGA_STRING_MODE_XM_STRING にセットしてください。ストリング自身は、どちらを選択したかによって、XmNxrtGaString または XmNxrtGaXmString でセットされます。

ストリングに値を表示

Text オブジェクトを使用して、XmNxrtGaValue リソースを持つ XRT/gauge オブジェクトの値を表示することができます。その場合、XmNxrtGaStringMode を XRTGA_STRING_MODE_STRING にセットし、XmNxrtGaValueObject を使用して、値を持つオブジェクトを Text オブジェクトにセットしてください。また、XmNxrtGaString の値として、"%g"などのフォーマットも指定してください。Text オブジェクトは指定されたオブジェクトから値を取り出し、設定されたフォーマットに従って、テキスト・ボックス内にその値を表示するようになります。

Counter、Indicator、Item、Needle、Switch などを value オブジェクトとすることができます。値オブジェクトは、値が変更された時は、オブジェクトに再表示を自動的に指示します。

テキストの行揃え

Text オブジェクト内のテキストは、XmNxrtGaAlignment を XRTGA_ALIGNMENT_CENTER、XRTGA_ALIGNMENT_END、XRTGA_ALIGNMENT_BEGINNING とセットすることにより、囲まれたボックス内で中央、右、左揃えにセットすることができます。

Text の配置

Text オブジェクトは、テキスト・ボックスの左上角の pixel 座標を指定することによりウィジェット上のどこにでも配置することができます。

下記のコードは、Circular Gauge ウィジェットに Text を追加している例です。

```
text = XtVaCreateWidget("text",
          xmXrtGaTextObjectClass,  cgauge,
          XmNx,                    100,
          XmNy,                    50,
          XmNxrtGaString,          "Hello World",
          XmNxrtGaAlignment,       XRTGA_ALIGNMENT_CENTER,
          COLOR_RES(XmNforeground,"blue"),
          COLOR_RES(XmNbackground,"wheat"),
          NULL);
MtCenterText(cgauge,text);
```

MtCenterText は、Text オブジェクトをウィジェット内で水平方向に中央に配置する便利な方法です。この方法は、*$XRTHOME/src/tools/mtgauge.c*.にあります。

20.3 サンプル・コード

下記のコードは、Needle オブジェクトの値が表示されるように Text オブジェクトをセットしている例です。%7.2 フォーマット指定子には引数がありませんが、オブジェクトは、Needle の現在値が使用されることを認識しています。

```
Widget needle; /* A needle whose value is displayed in the text */
/* Create a basic text area as a label */
text = XtVaCreateWidget("text",
          xmXrtGaTextObjectClass,  parent,
          XmNy,                    10,
          XmNxrtGaValueObject,     needle,
          XmNxrtGaStringMode,      XRTGA_STRING_MODE_STRING,
          XmNxrtGaString,          "Needle value: %7.2f",
          XmNxrtGaAlignment,       XRTGA_ALIGNMENT_CENTER,
          COLOR_RES(XmNbackground, "grey80"),
          COLOR_RES(XmNforeground, "black"),
          NULL);
```

20.4 リソースの一覧

Text オブジェクトの全てのリソースの一覧です。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。

XmNbackground **Pixel** ***inherited*** **CSG**

オブジェクトの背景の色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNfontList **XmFontList** **“fixed”** **CSG**

オブジェクトのテキストに使用されるフォント。

XmNforeground **Pixel** ***inherited*** **CSG**

オブジェクトの前景の色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。この色は、オブジェクトのテキストや線に使用されます。

XmNheight **Dimension** ***dynamic*** **G**

このリソースは、Text の縦のサイズによって決定されるので read-only です。

XmNwidth **Dimension** ***dynamic*** **G**

Pixel で表された Text オブジェクトの現在の幅。

XmNx **Position** **0** **CSG**

オブジェクトの左上角の x 座標。

XmNxrtGaAlignment **XrtGaAlignment** **XRTGA_ALIGNMENT_CENTER** **CSG**

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、`XRTGA_ALIGNMENT_BEGINNING`、`XRTGA_ALIGNMENT_CENTER`、`XRTGA_ALIGNMENT_END` です。

XmNxrtGaAllowWidgetPositioning **Boolean** **False** **CSG**

ウィジェットがテキスト領域の位置を書き換えることを許可します。このリソースが `False` であると、ユーザが位置をコントロールします。このリソースが `True` で Text オブジェクトの

XmNxrtGaValueObject が XrtGaItem 型のオブジェクトを指している場合は、Item オブジェクトがテキスト領域の位置を変更できるようになり、領域は、Item の XmNxrtGaRadius に従って配置されます。

XmNxrtGaBorder **XrtGaBorder** **XRTGA_BORDER_NONE** **CSG**

オブジェクトの枠線の種類。枠線の幅は、XmNxrtGaBorderWidth で指定されます。

XRTGA_BORDER_NONE	枠線なし
XRTGA_BORDER_PLAIN	前景色で描かれた実線
XRTGA_BORDER_3D_IN	沈んだように見える実線
XRTGA_BORDER_3D_OUT	浮いたように見える実線
XRTGA_BORDER_ETCHED_IN	沈んだように見える二重線
XRTGA_BORDER_ETCHED_OUT	浮いたように見える二重線
XRTGA_BORDER_FRAME_IN	沈んだように見える影つきのフレーム
XRTGA_BORDER_FRAME_OUT	浮いたように見える影付きのフレーム
XRTGA_BORDER_BEVEL	領域を囲むフレーム
XRTGA_BORDER_SHADOW	領域に対する影

XmNxrtGaBorderWidth **Dimension** **2** **CSG**

Pixel で表されたオブジェクトの枠線の幅。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaImage **XrtGaImage** **NULL** **CSG**

Text オブジェクトの背景に表示されるイメージ。

XmNxrtGaImageLayout **XrtGaImageLayout** **XRTGA_IMAGE_LAYOUT_CENTERED** **CSG**

背景のイメージの配置を指定します。有効な値は、XRTGA_IMAGE_LAYOUT_CENTERED または XRTGA_IMAGE_LAYOUT_TILED です。

XmNxrtGaImageMinimumSize **Boolean** **False** **CSG**

背景のイメージが存在する場合、そのイメージのサイズが Text オブジェクトの最小サイズになるよう指定します。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Text オブジェクトの表示、非表示をコントロールします。

XmNxrtGaString **char *** **NULL** **CSG**

Text オブジェクトに表示されるストリング。このリソースは複数行のテキストをサポートします。XmNxrtGaStringMode が XRTGA_STRING_MODE_STRING の時は、このリソースを使用してください。

XmNxrtGaStringMode **XrtGaStringMode** **XRTGA_STRING_MODE_STRING** **CSG**

Text オブジェクトに使用されるストリングの種類。有効な値は、XRTGA_STRING_MODE_STRING か XRTGA_STRING_MODE_XM_STRING です。XRTGA_STRING_MODE_STRING の場合は、通常のストリングが使用され、XmNxrtGaString がストリングを指します。XRTGA_STRING_MODE_XM_STRING の場合は XmString が使用され、XmNxrtGaXmString が XmString を指します。

XmNxrtGaValueObject **XrtGaObject** **NULL** **CSG**

セットされる値オブジェクト。NULL でなければ、Text オブジェクトは、XmNxrtGaValue リソースを使用して、このオブジェクトに値を問い合わせます。この値は、Text オブジェクトによって使用されるようになり、XmNxrtGaString リソース内のフォーマットされた値は置き換えられ、*printf* の演算子のような働きをします。

XmNxrtGaXmString **XmString** **NULL** **CSG**

Text オブジェクトに表示されるストリング。このリソースは、複数行のテキストをサポートします。XmNxrtGaStringMode が XRTGA_STRING_MODE_XM_STRING の場合は、このリソースを使用してください。

XmNy **Position** **0** **CSG**

オブジェクトの左上角の y 座標。

20.5 手続きと方法

XmCreateXrtGaText()

XrtGaText オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaText (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaText オブジェクトの親、*name* は新しく作成された XrtGaText オブジェクトの名前です。*arglist* は、作成時に XrtGaText に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaText オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaText()

```
XmIsXrtGaText (w)
```

wがXrtGaTextオブジェクトまたは、XrtGaTextオブジェクトのサブクラスの時は、Trueを返すマクロです。

21

Tick オブジェクト

XrtGaTick ■ Tick オブジェクトのリソースについて
サンプル・コード ■ リソースの一覧
手続きと方法

21.1 XrtGaTick

Tick オブジェクトは、Scale 上に均等に配置されるマークです。Tick マークには多くのリソースがありますので、配置や、形、数、インターバル、ラベル、ラベルに表示される小数点以下の桁数など、多くの属性をコントロールすることができます。

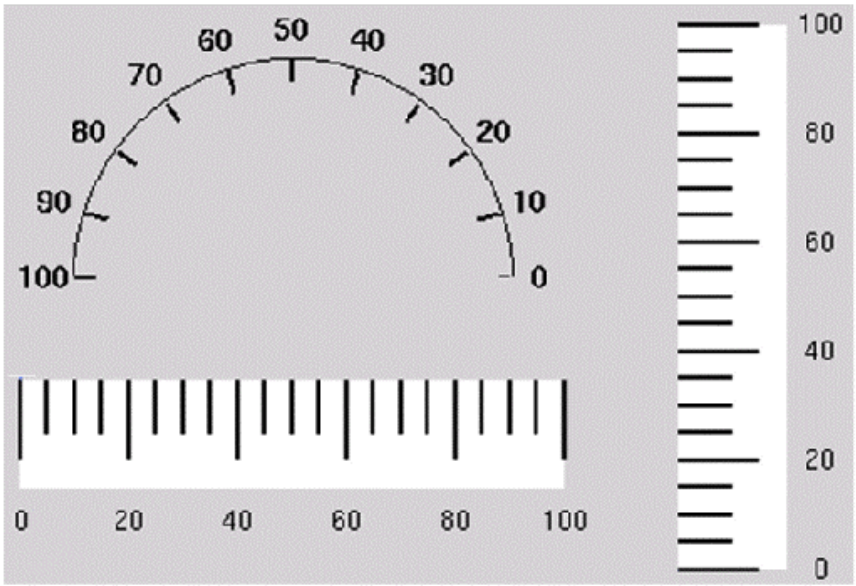


図38　Tick オブジェクトとそのラベル

Tick マークの開始と終了のポイントは任意に指定することができますので、Scale の特定な部分に Tick マークを配置することができます。また、オフセットをセットして Scale の開始ポイントの前または終了ポイントの後に Tick マークを表示させることもできます。

21.2 Tick オブジェクトのリソースについて

Tick オブジェクトを Scale にセットする

Tick オブジェクトには、それが表示される Scale がセットされている必要があります。XmNxrtGaScale を使用して、Circular または Linear Scale を Tick オブジェクトにセットしてください。

Tick オブジェクトの種類

Tick には、丸(circle)、ダイアモンド(diamond)、矩形(rectangle)、三角(triangle)の四つの種類が用意されています。どれを使用するかは、XmNxrtGaTickType で選択してください。

Tick オブジェクトの配置

Tick は、通常、Scale に沿って等間隔に配置されます。Tick オブジェクトは、セットされている Scale の *min* と *max* 値から、Tick 自身の *start*、*stop*、*increment* の値を決定します。XmNxrtGaPrecision がセットされていない場合は、デフォルトの精度が計算されます。XmNxrtGaDrawLabels が True になっていると、Scale がズームされたときにラベルの数が変更されます。(Scale が縮小されると、隣り合うラベルが重なってしまわないように、ラベルの数が減少し、Scale が拡大されれば、間隔が空き過ぎないようにラベルの数が増加します。)

Tick の間隔は、XmNxrtGaAutomatic を False にして独自に設定することもできます。XmNxrtGaStartValue で Tick の開始点、XmNxrtGaStopValue で終了点、XmNxrtGaIncrementValue でラベルのインクレメントをそれぞれ指定することができます。

Tick マークは、セットされている Scale の範囲の外に表示させることができます。この場合、どの程度範囲を拡張するかは、XmNxrtGaOffset でコントロールしてください。

Tick オブジェクトの大きさ

Tick マークの大きさは XmNxrtGaInnerExtent と XmNxrtGaOuterExtent でコントロールできます。ここで設定される値は、セットされる Circular または Linear Scale に対する割合です。例えば、XmNxrtGaInnerExtent が 1.0 で Circular Scale にセットされていれば、Tick は Scale の円周上から描画され、XmNxrtGaOuterExtent の値まで外向きに放射状に表示されます。XmNxrtGaOuterExtent も、セットされている Circular Scale の半径に対する割合です。

inner extent が outer extent より小さい場合、これらの値は入れ替わります。Iiner extent が outer extent と同じ値の場合は、tick は描画されません。目盛りのラベルのみを表示させたい場合は、この方法を使用することができます。XmNxrtGaAutomatic が False の場合、stop 値が start 値より小さいと、Tick もラベルも描画されません。

Tick マークの幅は、pixel で表され、XmNwidth で指定されます。

Tick のラベル

XmNxrtGaDrawLabels によってラベル表示をオンまたはオフにすることができます。自動モード(XmNxrtGaAutomatic が True)の場合、ラベルの間隔は XmNxrtGaPrecision の値で決定されます。ラベルの間隔をコントロールしたい場合は、このページの下の'精度に関する注意'を考慮しながら、XmNxrtGaPrecision を適する値に設定してください。XmNxrtGaPrecision をセットすると、自動的に XmNxrtGaPrecisionUseDefault は False にセットされます。

ラベルは XmNxrtGaLabelExtent で配置してください。このリソース値は、セットされる Scale の種類によって解釈のされ方が異なります。Circular Scale では、値が 1.5 の場合、ラベルは、Scale の半径の 1.5 倍分、外に配置されます。横向きの Linear Scale の場合、この距離は上方向から、縦向きの Linear Scale では左側から測定されます。

ラベルは、独自のファンクションを書くことによりカスタマイズすることができます。このファンクションのプロトタイプは、この章の最後のリソースの一覧にある、XmNxrtGaLabelFunc を参照してください。

精度(precision)に関する注意

XmNxrtGaPrecision の値は、Tick マークのラベル表示をコントロールします。このリソースをセットする場合には、次の三つの事に注意してください。

- XmNxrtGaPrecision の正の数は、表示される小数点以下の桁数を示します。例えば、XmNxrtGaPrecision が 3 にセットされると、ラベルは 0.001 の倍数(小数点以下第三位)で表示されます。
- XmNxrtGaPrecision の負の数は、表示される小数点の左側の桁数を示します。例えば、XmNxrtGaPrecision が-3 にセットされると、ラベルは 1000 の倍数で表示されます。
- XmNxrtGaPrecision に 0 がセットされると、ラベルは整数で表示されます。

このリソースの設定によっては、全く異なるスケールが表示されてしまいます。例えば、Tick オブジェクトの開始値が-25、終了値が 160 の時に、デフォルトで計算された精度が-1 とします。この場合、値-25 は-20 に切り捨てられてしまいます。精度を 0 に指定すれば、この桁は切り捨てまたは切り上げされなくなります。

21.3 サンプル･コード

下記のコードでは、Tick オブジェクトのほとんどのリソースがセットされています。多くのリソースはデフォルトのままにしておくことができます。この例は、参考までに各リソースの設定例を示しているだけですので、実際にはここまでセットする必要はありません。

```
ticks = XtVaCreateWidget("ticks",
      xmXrtGaTickObjectClass,cgauge,
      XmNxrtGaScale,            scale,
      XmNxrtGaAutomatic,        False,
      XmNxrtGaStartValue,       DVAL(0.0),
      XmNxrtGaStopValue,        DVAL(100.0),
      XmNxrtGaIncrementValue,   DVAL(10.0),
      XmNxrtGaOffset,           DVAL(0.0),
      XmNxrtGaPrecisionUseDefault, False,
      XmNxrtGaPrecision,        0,
      XmNwidth,                 2,
      XmNxrtGaDrawLabels,       True,
      XmNxrtGaIsShowing,        True,
      XmNxrtGaLabelExtent,      DVAL(1.15),
      XmNxrtGaInnerExtent,      DVAL(0.9),
      XmNxrtGaOuterExtent,      DVAL(1.0),
      COLOR_RES(XmNbackground, "black"),
      NULL);
```

21.4 リソースの一覧

Tick オブジェクトの全てのリソースの一覧です。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

XmNbackground **Pixel** ***inherited*** **CSG**

Tick オブジェクトの背景色。このリソースがセットされていない場合は、親ウィジェットの前景色が使用されます。

XmNfontList **XmFontList** **“fixed”** **CSG**

このオブエクトに使用されるテキストのフォント。

XmNwidth **Dimension** **1** **CSG**

Pixel で表されたオブジェクトの一番広い部分の幅。この値が 0 の場合は、Tick の種類が何であっても、可能な限り細い線が描画されます。Tick が circle の場合は、このリソースは無視されます。

XmNxrtGaAutomatic **Boolean** **True** **CSG**

Tick を自動的に生成するかどうか指定します。このリソースが `True` の場合は、セットされている Scale の *min* と *max* 及び `XmNxrtGaPrecision` の値が、Tick に適する開始と終了ポイント、インクレメントを決定します。これらの値は、自動的に対応するリソースをアップデートし、

これらのリソースに対する変更は無視されます。(値は、自動的に設定された値にリセットされます。) このリソースが False の時は、XmNxrtGaStartValue、XmNxrtGaStopValue、XmNxrtGaIncrement の値は、それぞれユーザ側でセットすることができます。

XmNxrtGaDebug **Boolean** **False** **CSG**

Xt error データベースの警告やエラー・メッセージを XtAppWarningMessage にプリントするかどうか指定します。False の場合は、リソース変換に関するエラーのみがプリントされます。

XmNxrtGaDrawLabels **Boolean** **False** **CSG**

True の場合は、XmNxrtGaPrecision の値が Tick に対するラベルの描画に使用されます。

XmNxrtGaIncrementValue **double** ***dynamic*** **CSG**

Tick のインクレメント(増分)。自動モードの時は、この値は read-only です。この値が、0 以下の場合は、最初の Tick のみが描画されます。

XmNxrtGaInnerExtent **double** **1.0** **CSG**

各 Tick の inner (Circular Scale の中央または Linear Scale の左上に向かう方向の) extent。この値は、セットされている Circular Scale の XmNxrtGaRadiusSize 値または、Linear Scale の高さ(横向きの場合)か幅(縦向きの場合)に関連しています。例えば、Circular Scale でこの値が 0.9 の場合、各 Tick の inner extent は、Scale の半径の 0.9 倍分原点から離れた値になります。横向きの Linear Scale で inner extent が 0.2 の場合は、Scale の高さの 0.2 倍分、Scale の上部から離れた値になります。Scale の左上角の位置は、Scale の位置によって決定されます。

XmNxrtGaIsShowing **Boolean** **True** **CSG**

Tick マークの表示、非表示を指定します。

XmNxrtGaLabelExtent **double** **1.0** **CSG**

ラベルの extent(各ラベルの中央)を指定します。この値は、セットされている Circular Scale の XmNxrtGaRadiusSize 値または、Linear Scale の高さ(横向きの場合)か幅(縦向きの場合)に関連しています。Circular Scale でこの値が 1.1 の場合、各ラベルの extent は、Scale の半径の 1.1 倍分原点から離れた値になります。横向きの Linear Scale でこの値が 0.5 の場合は、Scale の高さの 0.5 倍分、Scale の上部から離れた値になります。

XmNxrtGaLabelFunc **XtFunction** **NULL** **CSG**

Tick オブジェクトが生成するラベルとは異なるラベルを作成するときに使用するファンクション。このファンクションは、各 Tick ラベルに対して呼び出されます。プロトタイプは次のようになります。

```
static char * label_func(o, value, label, x, y)
  XrtGaObject o;
  double value;
  char *label;
  Position x, y;
```

o は tick オブジェクトで、*value* は現在の Tick ラベルの値。*label* は描画されるラベル、*x* はラベルの中央の x 位置、*y* はラベルの中央の y 位置です。

XmNxrtGaOffset **double** **0.0** **CSG**

Tick が Circular Scale にセットされている場合は、このリソースは、Tick マークが開始される値のオフセットを指定します。この値によっては、Scale の範囲外に Tick をシフトさせることができます。オフセットは、左右どちらの方向にも Scale の範囲の一回り(*max-min*)を上限として設定することができます。Tick が Linear Scale にセットされている場合は、このリソースは意味を持ちません。

XmNxrtGaOuterExtent **double** **1.0** **CSG**

各 Tick の outer (Circular Scale からウィジェットの境界へまたは、横方向の Linear Scale の上部から下部に向かう方向の) extent。この値は、セットされている Circular Scale の `XmNxrtGaRadiusSize` 値または、Linear Scale の高さ(横向きの場合)か幅(縦向きの場合)に関連しています。例えば、Circular Scale でこの値が 1.1 の場合、各 Tick の outer extent は、Scale の半径の 1.1 倍分原点から離れた値になります。横向きの Linear Scale で outer extent が 0.8 の場合は、Scale の高さの 0.8 倍分、Scale の上部から下方向に離れた値になります。

XmNxrtGaPrecision **int** ***dynamic*** **CSG**

ラベルに表示される値の小数点以下の桁数を指定します。このリソースの値が 0 以下の場合、ラベルは、全て整数で表示されます。

このリソースの値に負の数が設定された場合、精度は 10 の倍数になります。例えば、`XmNxrtGaPrecision` が-2 に設定されると、全てのラベルは 100 の倍数で表されます。`XmNxrtGaPrecision` が 1 の場合は、全てのラベルは 0.1 の倍数で表されます。このリソースをセットすると、副作用で対応する `XmNxrtGaPrecisionUseDefault` が `False` にセットされます。

このリソースの設定を誤ると、異なった Scale が表示されてしまいます。例えば、Tick オブジェクトの開始が-25、終了が 160 の時にデフォルトの精度が-1 にセットされると、値-25 は-20 に切り上げられてしまいます。-25 のままにするには、この位が切り捨てまたは切り上げられないように、リソース値を 0 にセットしてください。

XmNxrtGaPrecisionUseDefault **Boolean** **True** **CSG**

このリソースが `True` の場合、ウィジェットまたはオブジェクトは実行時に `XmNxrtGaPrecision` に適する値を計算します。`XmNxrtGaPrecision` に明示的に値が指定されていれば、このリソースは `False` にセットされます。`XmNxrtGaPrecision` に値が提供されていない場合は、このリソースを `False` にセットすることはできません。

XmNxrtGaScale **XmNxrtGaScale** **NULL** **CSG**

Tick がセットされる Circular または Linear Scale。この値が `NULL` の時は、Tick オブジェクトは描画されません。

XmNxrtGaStartValue **double** ***dynamic*** **CSG**

一番最初の Tick の値。この値は、セットされている Scale の *min* と *max* 値の間でなければなりません。自動モードでは、この値は read-only です。

XmNxrtGaStopValue **double** ***dynamic*** **CSG**

一番最後に来る Tick の値。この値は、セットされている Scale の *min* と *max* 値の間でなければなりません。自動モードでは、この値は read-only で、最後に表示される Tick の値と同じです。

XmNxrtGaTickType **XrtGaTickType** **XRTGA_TICK_TYPE_RECTANGLE** **CSG**

Tick マークの形を指定します。有効な値は、XRTGA_TICK_TYPE_RECTANGLE、XRTGA_TICK_TYPE_CIRCLE、XRTGA_TICK_TYPE_DIAMOND、XRTGA_TICK_TYPE_TRIANGLE です。

21.5 手続きと方法

XmCreateXrtGaTick()

XrtGaTick オブジェクトを作成する、標準の Motif 形式のファンクションです。

```
Widget
 XmCreateXrtGaTick (
     Widget      parent,
     char *      name,
     ArgList     arglist,
     Cardinal    argcount
     )
```

parent は作成される XrtGaTick オブジェクトの親、*name* は新しく作成された XrtGaTick オブジェクトの名前です。*arglist* は、作成時に XrtGaTick に渡される XrtArgVal リソースと値のペアのリストで、これは NULL とすることもできます。*argcount* は、*arglist* のエレメントの数です。

この手続きは、新しく作成された XrtGaTick オブジェクトを返しますが、オブジェクトが作成されなかった時は NULL が戻ります。

XmIsXrtGaTick()

```
XmIsXrtGaTick (w)
```

w が XrtGaTick オブジェクトまたは、XrtGaTick オブジェクトのサブクラスの時は、True を返すマクロです。

22

ユーザ･インタラクションのプログラミング

XRT/gauge 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
Widget/Object インタラクションの開始 ■ Widget/Object インタラクションのアップデート
移動とズーム ■ トランスレーションとズームのアップデート
Counter オブジェクトのリセット･アクション ■ ウィンドウのリサイズ

この章では、XRT/gauge のユーザ･インタラクション機能(どのようにユーザが XRT/gauge 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。

22.1 XRT/gauge のユーザ･インタラクション

デフォルトで、ユーザは、Circular Gauge と Circular Dial の Scale の移動やズームを始め、Linear Gauge、Linear Dial、Percent ウィジェットと様々なインタラクションを行うことができます。また、LEDIcon ウィジェットでは、ユーザが直接 Icon をトグルしたり、Counter オブジェクトで数値をリセットしたりできるほか、Indicator、Needle、Switch オブジェクトでも多くのユーザ･インタラクションがサポートされています。

22.2 トランスレーションとアクションの概要

`XrtGaLEDNumerical` を除く全ての XRT/gauge は、デフォルトのトランスレーション･テーブル(定義表)を持っています。このテーブルでは、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、任意のトランスレーションをテーブルに追加することによりデフォルトのトランスレーションを書き換えることができます。この方法は、特定なインタラクションを不可にしたり、あるインタラクションを開始するマウスまたはキーボートのイベントを変更するときに便利です。LEDNumerical ウィジェットにはトランスレーションは定義されていません。

■ **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/gauge 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、その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。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。

インタラクションには、Needle、Switch、Indicator、Icon 等に対する Widget/Object インタラクション、トランスレーション、ズーム、ResetCounter、Map アクションの五つの種類があります。Map アクションを除く後の四つのインタラクションには、デフォルトのトランスレーションが登録されています。

22.3 Widget/Object インタラクションの開始

*ActionStart()*が呼び出され、ウィジェットまたはオブジェクトに対する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 Interaction コールバックがどこで呼び出されるかを示しています。

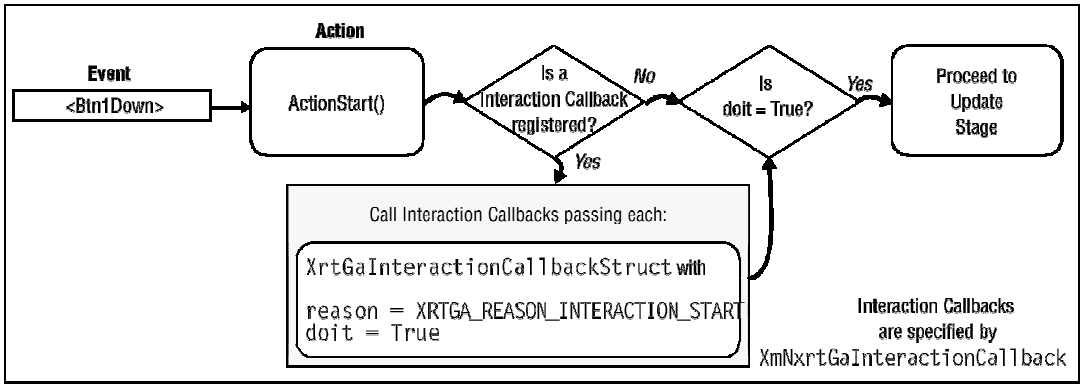


図39　ActionStart() アクション・ルーチン

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、XmNxrtGaInteractionCallback 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 */
  Boolean  doit; /* read/write when reason is not
                    XRTGA_REASON_INTERACTION_END. */
} XrtGaInteractionCallbackStruct;
```

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 XRTGA_REASON_INTERACTION_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、*ActionStart()*は、次の *ActionStart()*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。

22.4 Widget/Object インタラクションのアップデート

インタラクションがアップデートされようとすることをアプリケーションに通知するために、*ActionMove()*が呼び出されます。インタラクションが *ActionStart()*を問題なく通過した後、このアクションは、XRT/gauge ウィジェットをアップデートします。

*ActionMove()*は、XRT/gauge オブジェクトまたはウィジェットのインタラクションをアップデートします。このアクションは、Needle オブジェクト、Switch オブジェクト、Indicator オブジェクト、LEDIcon ウィジェットの XmNxrtGaValue をアップデートします。図 40 は、このアクション･ルーチンの図解で、追加された Interaction コールバックがどこで呼び出されるかを示しています。

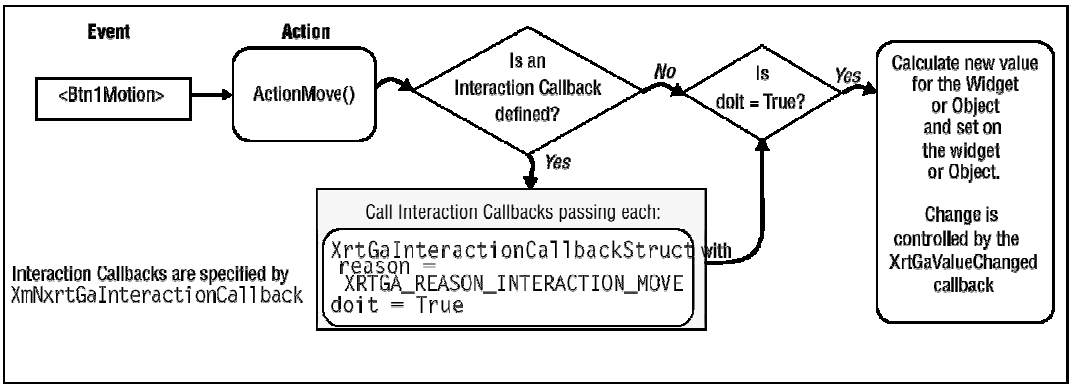


図40　ActionMove()アクション･ルーチン

インタラクションのコントロール

XmNxrtGaInteractionCallback を使用してトランスレーションやズーム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 */
      Boolean doit; /* read/write when reason is not
                       XRTGA_REASON_INTERACTION_END. */
} XrtGaInteractionCallbackStruct;
```

インタラクションを不可にするには、この構造体の *doit* パラメータを `False` にセットしてください。

22.4.1 Widget/Object インタラクションの終了

*ActionEnd()*が呼び出され、LEDIcon ウィジェットまたは Indicator オブジェクト、Needle オブジェクト、Switch オブジェクト上でのアクションが完了したことをアプリケーションに通知します。

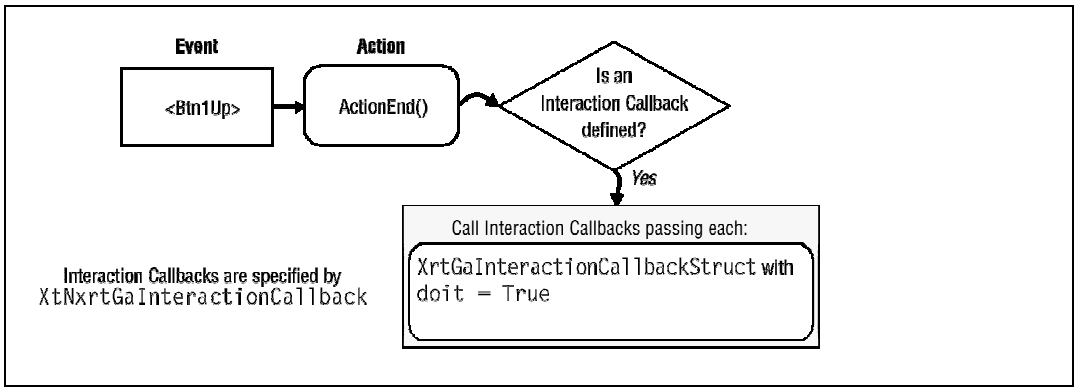


図41　ActionEnd() アクション・ルーチン

22.5 移動とズーム

TranslateStart()、*ZoomStart()*が呼び出され、Scale に対する移動またはズーム操作が開始されたことをアプリケーションに通知します。図 42 は、このアクション・ルーチンの図解で、追加された Transform コールバックがどこで呼び出されるかを示しています。

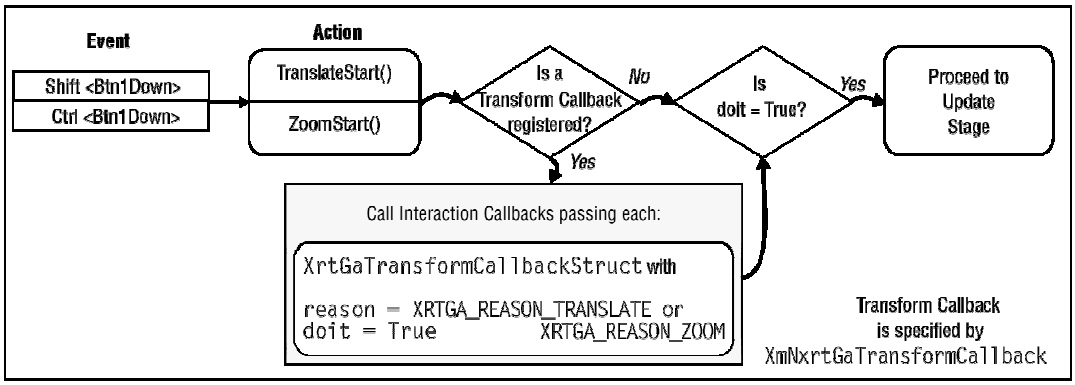


図42　ActionStart()アクション・ルーチン

移動とズームの監視と不許可

XmNxrtGaTransformCallback を使用して、移動やズームのプロセスを監視、制限したり、不許可にしたり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 */
  Dimension    new_size /* New radius size
                        for circular scales. */
  Dimension    new_height /* new height for linear scales */
  Dimension    new_width /* new width for linear scales */
  Position     new_x /* new x location*/
  Position     new_y /* new y location*/
  Boolean      doit /* True, Set to False
                        to disallow the operation */
} XrtGaTransformCallbackStruct
```

reason は、コールバックがどの時点で呼び出されるかを示しています。*reason* の値に関しては、P222 の XrtGaReason に関する説明を参照してください。*event* はイベントを指定します。*params* は、アクションに渡されるパラメータのリストで、*nparams* は、*params* リストにあるパラメータの数を指します。*new_size* は、Circular Scale の半径の新しい値、*new_height* と *new_width* は、Linear Scale の幅と高さの新しい値です。*new_x* は、新しい位置(Circular Scale の原点、Linear Scale の左上角)です。*doit* は、操作を許可または不許可にします。注意:　パラメータが-1 の場合は、値が変更されないか、現在のアクションでは使用されないことを意味します。

エンド・ユーザが Scale を移動またはズームしようとすると XmNxrtGaTransformCallback が呼び出されます。このコールバックは、*reason* が XRTGA_REASON_ZOOM、XRTGA_REASON_TRANSLATE_MOVE にセットされた *ZoomStart()*、*TranslateStart()*、*ZoomMove()*、*TranslateEnd()*アクションによって呼び出されます。

22.6 トランスレーションとズームのアップデート

*TranslateUpdate()*と *ZoomUpdate()*は、Scale に対する移動またはズーム操作がアップデートされようとすることをアプリケーションに通知するために呼び出されます。インタラクションが *TranslateStart()*、*ZoomStart()*アクションを正しく通過しないと、これらのルーチンは、XRT/gauge をアップデートしません。

22.6.1 トランスレーションとズーム

*TranslateUpdate()*アクションは、Circular または Linear Scale とその他の XRT/gauge ウィジェットとのインタラクティブなトランスレーションをアップデートします。同様に、*ZoomUpdate()*は、これらの Scale に対するインタラクティブなズームをアップデートします。これらのルーチンは、Scale のリソースを変更します。図 43 はこれらのアクションの図解で、追加した Transofrm コールバックがどこで呼び出されるかを示しています。

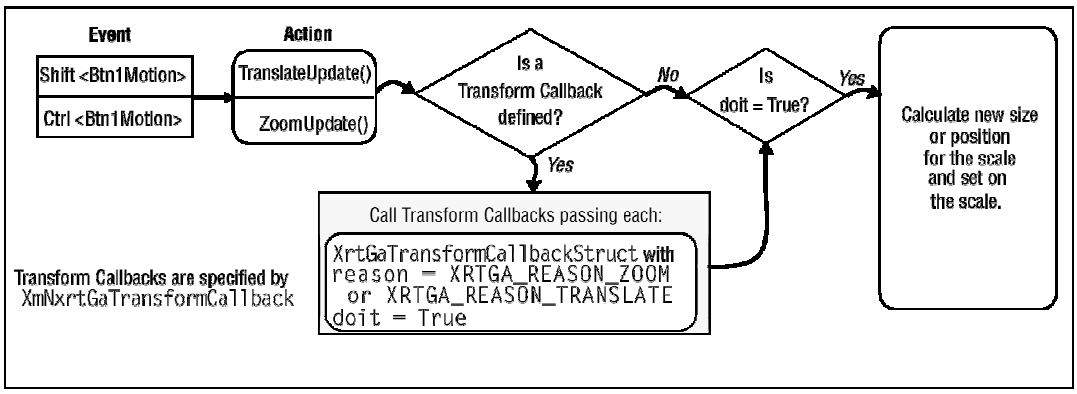


図43　TranslateUpdate() またはZoomUpdate() アクション・ルーチン

インタラクションのコントロール

XmNxrtGaTransformCallback を使用してトランスレーションやズームをコントロールすることができます。この構造体は、XmNxrtGaTransformCallback に渡されます。パラメータの値が-1 の場合は、値が変更されていないか、またはこれから変更されないことを意味します。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 */
      Dimension    new_size; /* New radius size
                        for circular scales */
      Dimension    new_height; /* New height for linear scales */
      Dimension    new_width; /* New width for linear scales */
      Position     new_x; /* New X location for both
                        linear and circular scales */
      Position     new_y; /* New Y location for both
                        linear and circular scales */
      Boolean      doit; /* Default: True. User sets to
                        False to disallow the operation */
} XrtGaTransformCallbackStruct;
```

インタラクションを不可にするには、構造体の *doit* パラメータを `False` にセットしてください。

22.6.2 トランスレーションとズーム

*TranslateEnd()*と *ZoomEnd()*は、ウィジェットまたはオブジェクトがアップデートされたことをアプリケーションに通知するために呼び出されます。

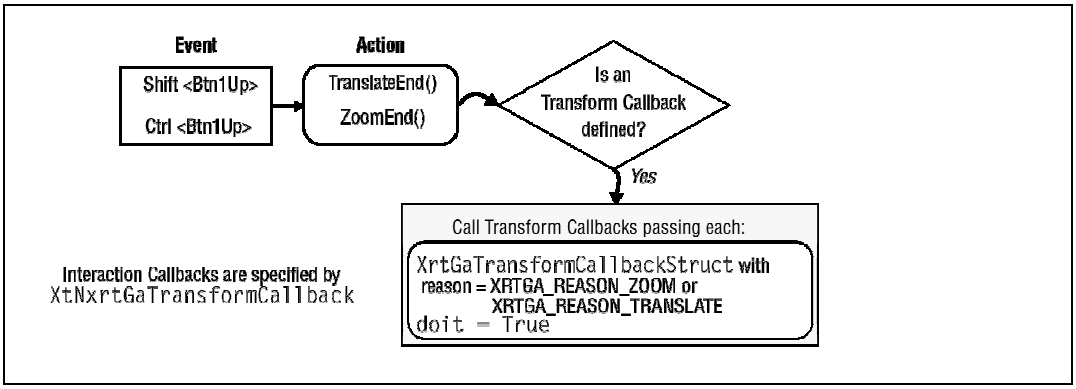


図44　ActionEnd() アクション・ルーチン

22.7 Counter オブジェクトのリセット・アクション

*ResetCounter()*アクションは、エンドユーザが Circular Gauge または Circular Dial 上で *Btn1Down* または *Btn1Up* を押したことをアプリケーションに通知する時に呼び出されます。図 45 は、このアクションの図解で、追加したインタラクション・コールバックがどこで呼び出されるかを示します。このアクションは Counter のリセットに対して呼び出されます。

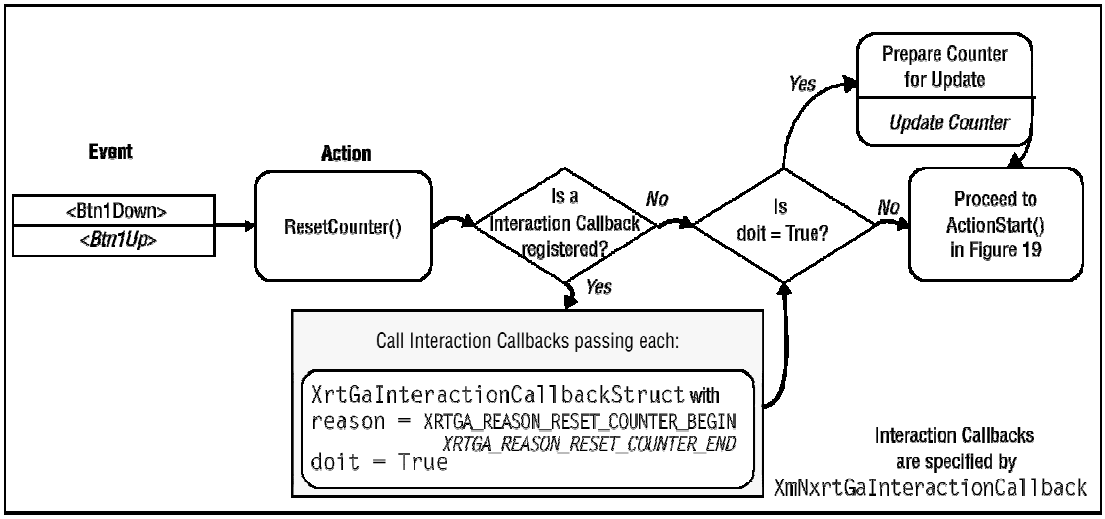


図45　ResetCounter() アクション・ルーチン

ResetCounter() は、Begin または End パラメータによって呼び出されます。*ResetCounter(End)*は、*Btn1Down* の代わりに *Btn1Up* で呼び出されることを除けば、他の動作は*ResetCounter(Begin)*と同様です。どちらも図45に示されており、*ResetCounter(End)* に付随するアイテムは斜体で表されています。

*ResetCounter(End)*は、*ResetCounter(Begin)*と同じリセット・ボタンでなければなりません。*Btn1* イベントが、同じウィジェットの Needle や Switch に影響を及ぼさないようにするには、*reason* が XRTGA_REASON_RESET_COPUNTER_BEGIN の時にこれらのオブジェクトのインタラクションを不許可にし、*reason* が XRTGA_REASON_RESET_COUNTER_END の時に元に戻してください。*reason* が XRTGA_REASON_RESET_COUNTER_END の時に *doit* を False にすると、リセット・ボタンを不可にすることができます。

22.7.1 マッピング

XRT/gauge のマッピング機能は、*XrtGaMap()*、*Map()*、XmNxrtGaMapCallback から成り立っています。XmNxrtGaMapCallback は、XrtGaMapCallbackStruct の助けを使用して *Map()*アクションが提供する値を保持します。フィールドのうち一つは *map*、XrtGaMapResult のポインタです。XrtGaMapResult 構造体は、*Map()*アクションの結果、*XrtGaMap()*が呼び出されたときにセットされます。この構造体は、マウスがクリックされた時にどの pixel にあり、対象になる Scale でどの値にその pixel が map されるかをアプリケーションに伝えます。

Map ファンクション

*XrtGaMap()*は、pixel 座標から Circular または Linear Scale 座標にマップします。このファンクションは、Scale オブジェクト、(x, y)pixel 座標、map 構造体のポインタを取り、Scale に基づいてウィジェットの座標を計算し、pixel 座標と Scale 座標を map 構造体に書き込みます。

```
Boolean
XrtGaMap(
      XrtGaObject      scale,
      Position         pix_x,
      Position         pix_y,
      XrtGaMapResult   *map
      )
```

Scale が有効な場合は、*XrtGaMap()*は True を返します。

XrtGaMapResult 構造体は次のように定義されます。

```
typedef struct {
  int pix_x /* The x-pixel value */
  int pix_y /* The y-pixel value */
  double value /* The scale value that corresponds
                  to (pix_x, pix_y) */
  double extent /* The extent that corresponds to
                   (pix_x, pix_y) */
  int x_offset /* The number of pixels that
                  (pix_x, pix_y) is offset from the
                  center or top left of the scale
                  in the x direction */
```

```
    int y_offset /* The number of pixels that
                    (pix_x, pix_y) is offset from the
                    center or top left of the scale
                    in the y direction */
    double angle /* The angle that corresponds to
                    (pix_x, pix_y) - only used for
                    circular scales*/
  } XrtGaMapResult
```

フィールドは、次のようになります。

pix_x	*XrtGaMap()*に渡される x 座標
pix_y	*XrtGaMap()*に渡される y 座標
value	Scale の(piz_x, pix_y)値
extent	Scale の半径、幅、高さの extent
angle	Circular Scale の角度
x-offset	Scale の原点から X 軸に沿った距離
y-offset	Scale の原点から Y 軸に沿った距離

XrtGaUnmap()

*XrtGaUnmap()*は、*XrtGaMap()*とは反対の方法です。この方法は、XRT/gauge ウィジェットの座標から pixel 座標へマップします。詳細は、P216 の *XrtGaUnmap()*を参照してください。

Map()アクション

*Map()*アクションを利用すると、子オブジェクトが Circular または Linear Scale のときに、ウィンドウ座標を取り、それを Scale 値に変換することができます。このアクションにはデフォルトではトランスレーションは定義されていません。トランスレーション・テーブルに *Map()*アクションを追加するには、次の例を参考にしてくだし。

```
static char translations[] = "\
  <Btn1Up>:        Map()";
XtOverrideTranslations(widget,
      XtParseTranslationTable(translations));
```

また、*$XRTHOME/src/gauge/examples/progress.c* には、Map コールバックを使用した例があります。

*Map()*アクションは、*XrtGaMap()*を呼び出して、ユーザ・イベントの pizel 座標にマップするウィジェット座標を決定します。このアクションは、XmNxrtGaMapCallback に渡される XrtGaMapResult を作成します。*Map()*を使用して、ユーザがクリックした場所の(x, y)値をみつけることができます。*Map()*の働きは、図 46 に示してあります。

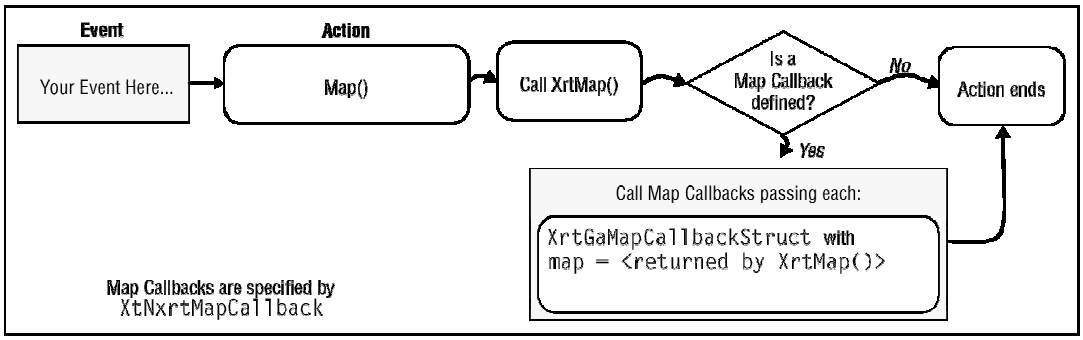


図46　*Map()*アクション・ルーチン

Map コールバック

*Map()*の結果を受けるには、XtNxrtGaMapCallback を使用してコールバックを登録する必要があります。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 */
     XrtMapResult     map; /* Read-only */
} XrtGaMapCallbackStruct;
```

この構造体のフィールドを変更しても、何も効果はありません。

Circular Gauge、Circular Dial、Linear Gauge、Linear Dial、Percent ウィジェットの XmNxrtGaMapCallback は、XRT/gauge ウィジェットに *Map()*が登録された時、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。XrtGaMapCallbackStruct のポインタが各 Map コールバックに渡されます。

下記の例では、Map アクションには、三つのマウス操作が指定され、ユーザによって書かれた *MapCB()*ファンクションによって、このアクションが開始されています。コールバック・ファンクションは、二つの関連するウィジェット(Circular Gauge と Linear Gauge 等)のオブジェクトの値をシンクロナイズさせるために使用されています。この方法により、両方のウィジェットは常に同じ値を表示するようになります。尚、このコードは、名前によってアイテムをリストから取得する *GetObjectByName()*が前もって書かれていることを前提としています。

```
static char translations[] = "\
     <Btn1Up>:        Map()
     <Btn1Motion>:    Map()
     <Btn1Down>:      Map()";

Widget widget; /* A widget in the XRT/gauge family */
XtOverrideTranslations(widget,
                  XtParseTranslationTable(translations));
XtAddCallback(widgetObject,
     XmNxrtGaMapCallback,
     (XtCallbackProc)MapCB,
     (XtPointer) anotherWidgetsObject);
```

```
MapCB(w, client_data, cbs)
     Widget                    w;
     XtPointer                 client_data;
     XrtGaMapCallbackStruct    *cbs;
 {
     char                      string[20];
     Widget                    anotherWidgetsObject;
     XrtGaObject               object;
     XrtGaList                 list;

     anotherWidgetsObject = (Widget) client_data;
     sprintf(string, "%.0f%%", cbs->map.value);

     /* Set Values on both progress charts */
     XtVaGetValues(w,
         XmNxrtGaItemList,        &list,
         NULL);
 object = GetObjectByName(list, ITEM_NAME);
 if (object != NULL) {
     XtVaSetValues(object,
         XmNxrtGaValue,   DVAL(cbs->map.value),
         NULL);
 }

 XtVaGetValues(w,
         XmNxrtGaTextList,        &list,
         NULL);
 object = GetObjectByName(list, TEXT_NAME);
 if (object != NULL) {
     XtVaSetValues(object,
         XmNxrtGaString,          string,
         NULL);
 }

 XtVaGetValues(anotherWidgetsObject,
         XmNxrtGaItemList,        &list,
         NULL);
 object = GetObjectByName(list, ITEM_NAME);
 if (object != NULL) {
     XtVaSetValues(object,
         XmNxrtGaValue,       DVAL(cbs->map.value),
         NULL);
}
 XtVaGetValues(anotherWidgetsObject,
         XmNxrtGaTextList,        &list,
         NULL);
 object = GetObjectByName(list, TEXT_NAME);
 if (object != NULL) {
     XtVaSetValues(object,
         XmNxrtGaString,          string,
         NULL);
 }
}
```

22.7.2 値変更のコールバック

値変更のコールバックは次のように定義されています。

```
typedef struct {
  int          reason; /* read-only */
  Xevent       *event; /* read-only */
  Double       value
  Boolean      doit; /* initially True */
} XrtGaValueChangedCallbackStruct;
```

このコールバックは、Item、Indicator、Needle、Switch オブジェクトで値が変更されたときに呼び出されます。新しい値は `value` です。この値は、*reason* が `XRTGA_REASON_VALUE_CHANGE_BEGIN` の時に変更したり、*doit* を `False` にして値が変更されないようにすることができます。

reason が `XRTGA_REASON_VALUE_CHANGE_END` の時にこのコールバックが呼び出されると、ウィジェットは値変更が完了した通知のみを出します。

22.7.3 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/gauge のプログラムは、このメカニズムを利用して、XRT/gauge 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。XRT/gauge 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_gauge;
XtAddEventHandler( my_gauge,
  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。

22.8 ウィンドウのリサイズ

アプリケーションの多くは、XRT/gauge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/gauge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/gaugeウィジェットの最適なサイズを決定し、XRT/gauge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、XtNxrtGaExposeCallback、XtNxrtGaResizeCallbackを使用してコールバックを指定することにより、XRT/gauge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XrtGaCallbackStructを指します。詳細はP224を参照してください。リサイズのコールバックは、XRT/gauge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、XRT/gauge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XRT/gauge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/gauge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(mygauge, XmNxrtGaResizeCallback,
  resize_proc, &my_data);
```

下のリサイズ・コールバックは、ウィジェットがリサイズされた時にCircular Gaugeウィジェットの Scale を中央に配置します。

```
/* Callback to center a Circular Gauge Widget's scale */
void
resizeCB(w, client_data, call_data)
      Widget       w;
      XtPointer    client_data;
      XtPointer    call_data;
{
      Dimension    width, height;
      XrtGaObject  circular_scale;

      XtVaGetValues(w,
          XmNwidth,                &width,
          XmNheight,               &height,
          XmNxrtGaCircularScale,   &circular_scale,
          NULL);
      if (!circular_scale) {
          return;
        }
```

```
        XtVaSetValues(circular_scale,
            XmNxrtGaXOrigin,         width/2,
            XmNxrtGaYOrigin,         height/2,
            NULL);
    }
```

23

さらに進んだ使用法

独自の描画ファンクション ■ XtConvertAndStore にイメージをセットする

23.1 独自の描画ファンクション

Indicator、Marker、Needle、Switch オブジェクト、Linear Dial ウィジェットには、独自の描画ファンクションを提供することができます。ユーザによって定義されたファンクションは次のプロトタイプを持っています。

```
void
draw_func(object, user_data)
XrtGaObject object;
XtPointer user_data;
```

このファンクションは、指定されたオブジェクトをウィジェットが描画する時に呼び出されます。この呼び出しでは、user_data パラメータを通じてウィジェットから得られた XrtGaDrawData に対するポインタを取得することができます。

```
typedef struct {
  Position     x; /*X position of drawing origin */
  Position     y; /*Y position of drawing origin */
  Dimension    width; /*Width of draw object */
  Dimension    height; /*Height of draw object */
  Double       angle; /*Angle at which object is to be drawn */
  Position     x_offset; /*x distance from (x, y) */
  Position     y_offset; /*y distance from (x, y) */
} XrtGaDrawData;
```

オブジェクトにセットされたリソースに基づいて、ウィジェットは、オブジェクトの位置、大きさ、角度を計算します。これらの情報は、XrtGaDrawData 構造体に渡されます。各パラメータが何を意味するかは、オブジェクトによって異なります。(P184 参照)

ユーザは、この情報を使用してオブジェクトを描くことができます。また、*XrtGaGetUserDrawInfo()*を呼び出して、描画に関わるウィジェット特有の情報(ウィジェットが現在描画している Window または Pixmap 等)を得ることができます。

XrtGaDrawData 構造体の情報に基づいて、スケーリング、回転、位置決めなどが指定できるポリゴンを描画することができます。この描画メカニズムを使用するには、XrtGaPolyDrawStruct 構造体をセットし、*XrtGaDrawPolyTransform()*に渡す必要があります。XrtGaPolyDrawStruct は次のように定義されています。

```
typedef struct {
 int          operations;  /* Bitwise OR of XRTGA_* operations */
 Position     x;           /*X position to translate to. */
 Position     y;           /*Y position to translate to. */
 Dimension    width;       /*Width to scale to. */
 Dimension    height;      /*Height to scale to. */
 Double       angle;       /*Angle to rotate to (0 degrees is E) */
 Xpoint       *shape;      /*List of points */
 int          npoints;     /*Number of points in "shape". */
} XrtGaPolyDrawStruct;
```

多くの場合、XrtGaDrawData 構造体の情報を XrtGaPolyDrawStruct にコピーすることができますが、ユーザは、ポリゴンの回転を定義する必要があります。その場合、ポリゴンの原点は、ポリゴンの回転の中心です。ポリゴンの x 方向は、XrtGaPolyDrawStruct に指定される幅、y 方向の範囲は XrtGaPolyDrawStruct に指定される高さにスケーリングされます。指定された角度は東から始まり時計回りに回転します。Y 方向は画面の下方向を指します。x と y フィールドは、ポリゴンが移動されるウィジェット内での位置を指定します。

トランスフォーメーションの順序は、スケーリング、回転、移動です。ユーザは、オペレーション・フラグを指定して、どのトランスフォーメーションを行うかをコントロールすることができます。下記のトランスフォーメーションの定義を bitwise OR することにより、組み合わせを指定することができます。

```
#define XRTGA_TRANSFORM_NONE 0x00
#define XRTGA_TRANSLATE 0x01
#define XRTGA_ROTATE 0x02
#define XRTGA_SCALE 0x04
#define XRTGA_TRANSFORM_ALL (XRTGA_TRANSLATE | XRTGA_ROTATE |
                             XRTGA_SCALE)
```

*XrtGaDrawPolyTransform()*の呼び出しでは、正しいウィンドウまたは pixmap への描画、ポリゴンの背景色の設定など、描画に関する詳細はウィジェットが管理しますので、この方法を使用する場合は、*XrtGaUserDrawInfo()*を呼び出す必要はありません。

$XRTHOME/src/gauge/util/draw_func.c には、この方法の使用例が幾つかあります。これらのファンクションは、ウィジェットが定義済みのオブジェクトを描画するときに使用されています。

例

次の例では、XtVaSetValue のコールで、XmNxrtGaDrawFunc リソースの値としてファンクションに名前を付け、ウィジェットは、このファンクションを呼び出し、該当するパラメータをセットしています。

```
XtVaSetValues(needle,
      XmNxrtGaDrawFunc,DrawNeedle,
      NULL);
/* An example of a user-supplied drawing function */
```

```
DrawNeedle(object, draw, user_data)
      XrtGaObjectobject; /* One of those mentioned above */
      XtPointeruser_data;
{

      /*
      * Fancy arrow shape
      */
      static Point arrow[] = {{10, 0}, {0, 10}, {110, 3}, {100, 10},
          {130, 0}, {100, -10}, {110, -3}, {0, -10}};

      XrtGaPolyDrawStruct poly;
      XrtGaDrawData *dd;

      dd = (XrtGaDrawData *) user_data;
      poly.angle = -dd->angle;
      poly.width = dd->width;
      poly.height = dd->height;
      poly.x = dd->x;
      poly.y = dd->y;
      poly.operations =
          (XRTGA_TRANSLATE | XRTGA_ROTATE | XRTGA_SCALE);
      poly.shape = arrow;
      poly.npoints = XtNumber(arrow);
      XrtGaDrawPolyTransform(object, (XtPointer)&poly);
}
```

23.1.1 XrtGaDrawData 構造体のパラメータ

XrtGaIndicator

`x`	Indicator が描画される左上角の x 位置
`y`	Indicator が描画される左上角の y 位置
`width`	Indicator の長さ(x 方向 – `XrtGaDrawPolyStruct` の幅にマップ)
`height`	Indicator の幅(y 方向 – `XrtGaDrawPolyStruct` の高さにマップ)
`angle`	Scale が Indicator を描画しようとする角度。Y 方向は下向きなので、この角度は、*XrtGaDrawPolyTransform()*に提供される角度の逆になります。
`x_offset`	ゼロ
`y_offset`	ゼロ

角度は、Indicator の方向を意味します。

0 degrees	右
90 degrees	下
180 degrees	左
270 degrees	上

XrtGaMarker

x	セットされる Circular Scale の X 原点
y	セットされる Circular Scale の Y 原点
width	Marker の長さ(x-方向 -- XrtGaDrawPolyStruct の幅にマップ)
height	Marker の幅(y-方向 – XrtGaDrawPolyStruct の高さにマップ)
angle	Scale が Marker を描画しようとする角度。Y 方向は下向きなので、この角度は、*XrtGaDrawPolyTransform()*に提供される角度の逆になります
x_offset	Scale の原点から Marker の inner radius に対する x オフセット
y_offset	Scale の原点から Marker の inner radius に対する y オフセット

XrtGaNeedle

x	セットされる Circular Scale の X 原点
y	セットされる Circular Scale の Y 原点
width	Needle の長さ(x-方向 -- XrtGaDrawPolyStruct の幅にマップ)
height	Needle の幅(y-方向 – XrtGaDrawPolyStruct の高さにマップ)
angle	Scale が Needle を描画しようとする角度。Y 方向は下向きなので、この角度は、*XrtGaDrawPolyTransform()*に提供される角度の逆になります
x_offset	Needle は原点から開始されるのでゼロ
y_offset	Needle は原点から開始されるのでゼロ

XrtGaSwitch

x	セットされる Circular Scale の X 原点
y	セットされる Circular Scale の Y 原点
width	Scale の半径(x-方向 -- XrtGaDrawPolyStruct の幅にマップ)

height	Scale の半径(y-方向 – XrtGaDrawPolyStruct の高さにマップ)
angle	Scale が Switch を描画しようとする角度。Y 方向は下向きなので、この角度は、*XrtGaDrawPolyTransform()*に提供される角度の逆になります
x_offset	Switch は原点から開始されるのでゼロ
y_offset	Switch は原点から開始されるのでゼロ

注意: Switch の原点がポリゴン内にある時に、サイズを Scale の半径より大きくする場合は、そのポリゴンをスケーリングする必要があります。

draw_func.c の *XrtGaDrawSwitchPointer()*にこの例がありますので参照してください。

23.2 XtConvertAndStore でイメージをセットする

Fallback リソースでイメージをセットする代わりに、*XtConvertAndStore()*を使用することができます。手順は下記の例を参照してください。

1. 最初に、イメージがセットされるウィジェット及び、イメージの完全なパス名をパラメータとしてとるファンクションを作成します。

```
GetColorImage(w, name)
      Widget   w; /* The parent widget */
      char     *name; /* The full path and filename
                   for the image, which may include
                   environment variables. */
{
      char         file_name[MAXPATHLEN + 1];
      XrtGaImage   image;
      XrmValue     from, to;

      MtExpandFilename(name, file_name); /* Environment
                   variables are expanded here. */

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

      if (!XtConvertAndStore(w, XmRString, &from,
                                  XmRXrtGaImage,&to))
          {
          fprintf(stderr,
              "Conversion of %s to XmRXrtGaImage failed\n",
              file_name);
          image = NULL;
          }
          return(image);
}
```

2. 次にそのファンクションを呼び出します。

```
XrtGaImage image;
image = GetColorImage(cimage_ldial,
      "$XRTHOME/src/icons/teddybear_gift.jpg");
```

3. XtVaSetValues を使用してリソースに色をセットします。ここでは、Linear Dial ウィジェットのオンの色にセットされています。

```
XtVaSetValues(linear_dial_widget,
      XmNxrtGaOnImage, image,
      NULL);
XrtPDSImageDestroy(image);
```

XtVaSetValues への呼び出しはイメージをコピーしますので、*XtConvertAndStore()* の呼び出しで作成されたイメージは破棄してください。

JPEG イメージの変換ライブラリは、XRT/gauge には自動的に登録されません。下記のようにして、JPEG イメージ変換を登録し、XRT/gauge で使用できるようにしてください。

```
XrtGaEnableImageType((XtPointer)XrtPDSJpegImageUtils);
```

24

PDS ユーティリティ

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

24.1 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。

24.1.1 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/gauge 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、24.2.2 を参照してください。

24.2 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
XrtGaEnableImageType(
  XtPointer    class,
  XtPointer    functions
  )
```

class はウィジェット・クラス(i.e. xtXrtGaCircularGaugeClass)、*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;
  XrtPDSImageHeaderCompareProc check_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/XrtGauge.h>
#include <Xm/XrtPDSJpeg.h>
XrtGaEnableImageType((XtPointer)xtXrtGaCircularGaugeClass,
  (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 type 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。

24.2.1 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/gauge、XRT/graph、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 XrtPDSXpmImageUtils、JPEG は XrtPDSJpegImageUtils) 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。

24.2.2 任意のイメージ型

任意のイメージ型の読み書きは、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;
  XrtPDSImageHeaderCompareProc check_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。

```
XrtGaEnableImageType(xtXrtGaCircularGaugeWidgetClass,
                     TargaImageUtils)
```

これで、(XtNxrtGaImage 等の)リソースの Targa イメージを読み書きができるようになります。

24.2.3 イメージ・ファンクションを書く

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。

24.3 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。詳細はリリースノートを参照してください。

24.4 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)。

24.4.1 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。

24.4.2 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一つの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。

24.4.3 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でなければなりません。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。(ソートに関する詳細は 24.4.6 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()*を呼び出します。

```
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には残っています。アイテムを除去すると同時に破棄するには、*XrtPDSListDestroy()*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();
```

アイテムに対してデータ構造体が定義されており、その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。

24.4.4 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 24.4.6 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);
```

24.4.5 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。

*XrtPDSCmpString()*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、*XrtPDSCmpStringIgnoreCase()*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。

24.4.6 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、24.4.4 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。

24.4.7 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。

24.5 手続きと方法

24.5.1 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 X 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 bit 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。

24.5.2 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()* へ渡すことができます。

```
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()* へ渡すことができます。

```
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()*へ渡すことができます。

```
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()*へ渡すことができます。

```
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()*へ渡すことができます。

```
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なければなりません。

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なければなりません。

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 になっていますが、*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、み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参照しないでください。

```
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,
  XrtPDSListItemCompareProc     compare_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,
  XrtPDSListItemDestroyProc     destroy_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

マクロと方法

A.1 マクロ

次のマクロを利用して、色やダブル値をセットすることができます。これらは、*$XRTHOME/include/Xm/XrtGauge.h* に定義されています。

マクロ名	定義
COLOR_RES(res, color)	`XtVaTypedArg, res, XmRString, \ color, strlen(color) + 1`
DVAL(value)	`XrtGaDoubleToArgVal(value)`

A.2 方法

XrtGaArgValToDouble()

`XtArgVal` をダブルに変換します。これは、値を `XtArgVal` に変換する *XrtGaDoubleToArgVal()*の逆の手続きです。*value* は、変換される `XtArgVal` です。

```
double
XrtGaArgValToDouble(value)
  XtArgVal value;
```

XrtGaDoubleToArgVal()

ダブルを `XtArgVal` に変換します。これは、*XrtGaArgValToDouble()*とは逆の手続きです。この手続きは、`XtSetValues`、`XtVaSetValues` などのルーチンでダブル型のリソースをセットするときに使用されます。`dval` は、変換されるダブルです。

```
XtArgVal
XrtGaDoubleToArgVal(dval)
  double dval;
```

DVAL() をマクロとして使用することができます。その場合、*$XRTHOME/src/tools/mtgauge.h* をインクルードしてください。

XrtGaDrawPolyTransform()

ポイントのリストをポリゴンとして移動、スケーリング、回転し、描画します。w はトランスフォーメーションが起きるウィジェットです。data は、XtPointer にキャスト(cast)される XrtGaPolyDrawStruct 構造体のポインタが必要です。

```
void
XrtGaDrawPolyTransform(w, data)
  Widget       w;
  XtPointer    data;
```

描画ファンクションの定義に関する詳細は、P181 の解説を参照してください。

XrtGaEnableImageType()

イメージを変換するときに使用される新しいファンクションのセットを登録します。新しいファンクションの登録が成功すると True、そうでなければ False が戻ります。functions は、XrtPDSImageFuncs 構造体のポインタです。詳細は、P189 の XrtPDSImage に関する解説を参照してください。

```
Boolean
XrtGaEnableImageType(functions)
  XtPointer    functions;
```

XrtGaMap()

Scale のパラメータに基づいて、pix_x と pix_y を値と extent にマップします。成功すると(Scale が有効であると)True を返します。この場合の scale は Circular または Linear Scale を指します。pix_x は x pixel 位置、pix_y は y pixel 位置です。map は値、extent 等の情報が返される構造体です。

```
Boolean
XrtGaMap(scale, pix_x, pix_y, map)
  XrtGaObject      scale;
  Position         pix_x;
  Position         pix_y;
  XrtGaMapResult   *map;
```

XrtGaSetXmStringConverter()

フォーマットされた XmString 値を作成する、XmRXmString リソース変換プログラムをセットします。この手続きは、必ず、XRT/gauge ウィジェットが作成される前に呼び出してください。

```
void
XrtGaSetXmStringConverter(void)
```

XrtGaUnmap()

Scale のパラメータに基づいて、値と extent を位置(pix_x と pix_y)にマップします。位置は、XrtGaMapResult 構造体に返されます。成功すると(Scale が有効で値が *min* と *max* の間にあれば)True を返します。この場合 scale は Circular または Linear Scale を指します。value はマップに使用される Scale の値、extent は Circular Scale の放射線上の方向です。Linear Scale では、extent の方向は Scale の方向に対して

右側の角度で、Scale の幅または高さに関連しています。map は値、extent 等の情報が返される構造体です。

```
Boolean
 XrtGaUnmap(scale, value, extent, map)
     XrtGaObject      scale;
     double           value;
     double           extent;
     XrtGaMapResult   *map;
```

B

データ型

XRT/gauge のデータ型、列挙法、構造体のアルファベット順のリストです。

B.1 列挙法

XrtGaAlignment

Counter オブジェクトでは、XmNxrtGaResetAlignment で使用され、リセット・アイコンの配置を指定します(XRTGA_ALIGNMENT_CENTER はこの場合使用できません。)State と Text オブジェクトでは、XmNxrtGaAlignment で使用され、オブジェクトに関連するテキストの配置に使用されます。

```
XRTGA_ALIGNMENT_BEGINNING
XRTGA_ALIGNMENT_CENTER
XRTGA_ALIGNMENT_END
```

XrtGaBorder

XRT/gauge ウィジェット、Counter オブジェクト、Text オブジェクトの XmNxrtGaBorder に使用され、枠線の種類を指定します。

```
XRTGA_BORDER_NONE
XRTGA_BORDER_PLAIN
XRTGA_BORDER_3D_IN
XRTGA_BORDER_3D_OUT
XRTGA_BORDER_ETCHED_IN
XRTGA_BORDER_ETCHED_OUT
XRTGA_BORDER_FRAME_IN
XRTGA_BORDER_FRAME_OUT
XRTGA_BORDER_BEVEL
XRTGA_BORDER_SHADOW
```

XrtGaCalcMode

Percent ウィジェットの XmNxrtGaCalcMode に使用され、データの表示法を指定します。Percent モードでは、リストの各アイテムのパーセンテージの値が使用されます。($\frac{(value - \min)}{(\max - \min)} \times 100$)。Radio モードでは、リスト上の全てのアイテムの合計に対する

各アイテムの割合が使用されます。

```
XRTGA_CALC_MODE_PERCENT
XRTGA_CALC_MODE_RATIO
```

XrtGaDirection

LEDIcon ウィジェット、Circular Scale オブジェクト、Linear Scale オブジェクトの XmNxrtGaDirection に使用され、ウィジェットまたはオブジェクトの進行方向を指定します。

```
XRTGA_DIRECTION_FORWARD
XRTGA_DIRECTION_BACKWARD
```

XrtGaDisplayMode

LEDIcon の XmNxrtGaDisplayMode に使用され、表示モードを指定します。Magnitude モードでは、XmNxrtGaValue リソースが幾つのアイコンをオンにするか決定します。Binary モードでは、二進法で表された値がどのアイコンをオンにするか決定します。例えば、値が 9 であると、二進法では 1001 となるので、0 番目と 3 番目のアイコンがオンになります。

```
XRTGA_DISPLAY_MODE_MAGNITUDE
XRTGA_DISPLAY_MODE_BINARY
```

XrtGaFillPattern

塗りつぶしのパターンを指定する列挙法です。

```
XRTGA_FPAT_NONE
XRTGA_FPAT_SOLID
XRTGA_FPAT_25_PERCENT
XRTGA_FPAT_50_PERCENT
XRTGA_FPAT_75_PERCENT
XRTGA_FPAT_HORIZ_STRIPE
XRTGA_FPAT_VERT_STRIPE
XRTGA_FPAT_45_STRIPE
XRTGA_FPAT_135_STRIPE
XRTGA_FPAT_DIAG_HATCHED
XRTGA_FPAT_CROSS_HATCHED
```

XrtGaImageLayout

XRT/gauge ウィジェット、Counter、Frame、Range、Text オブジェクトの XmNxrtGaImageLayout に使用され、背景のイメージの配置を指定します。

```
XRTGA_IMAGE_LAYOUT_CENTERED
XRTGA_IMAGE_LAYOUT_TILED
```

XrtGaIndicatorType

Indicator オブジェクトの XmNxrtGaIndicatorType に使用され、Indicator の種類を指定します。

```
XRTGA_INDICATOR_TYPE_RECTANGLE
XRTGA_INDICATOR_TYPE_CIRCLE
XRTGA_INDICATOR_TYPE_TRIANGLE
XRTGA_INDICATOR_TYPE_POINTER
XRTGA_INDICATOR_TYPE_IMAGE
```

XrtGaInteractionType

LEDIcon ウィジェット、Indicator、Needle、Switch オブジェクトの XmNxrtGaInteractionType に使用され、ウィジェットまたはオブジェクトとマウスのインタラクションの種類を指定します。

オブジェクトの場合、有効な値は、NONE、RELATIVE_TO_MOUSE(オブジェクトとマウスは、インタラクションが開始された時と常に同じ距離にある)、SNAP_TO_MOUSE(オブジェクトは、インタラクションが開始されるとマウスの位置に移動する)です。

LEDIcon ウィジェットの場合、NONE にセットされればインタラクションはなし、その他の二つのの値にセットされればインタラクションあり、というように Boolean 値のように働きます。インタラクションは XmNxrtGaDisplayMode の値に依存します。

```
XRTGA_INTERACTION_TYPE_NONE
XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE
XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE
```

XrtGaLinePattern

State オブジェクトの XmNxrtGaLinePattern に使用され、線の種類を指定します。

```
XRTGA_LPAT_NONE
XRTGA_LPAT_SOLID
XRTGA_LPAT_LONG_DASH
XRTGA_LPAT_DOTTED
XRTGA_LPAT_SHORT_DASH
XRTGA_LPAT_LSL_DASH
XRTGA_LPAT_DASH_DOT
```

XrtGaMarkerType

Marker オブジェクトの XmNxrtGaMarkerType に使用され、Marker の種類を指定します。

```
XRTGA_MARKER_TYPE_CIRCLE
XRTGA_MARKER_TYPE_TRIANGLE
XRTGA_MARKER_TYPE_ARROW
XRTGA_MARKER_TYPE_IMAGE
```

XrtGaNeedleType

Needle オブジェクトの XmNxrtGaNeedleType に使用され、Needle の種類を指定します。

```
XRTGA_NEEDLE_TYPE_POINTER
XRTGA_NEEDLE_TYPE_TRIANGLE
XRTGA_NEEDLE_TYPE_ARROW
```

XrtGaOffsetType

Circular Dial ウィジェットの XmNxrtGaOffsetType に使用され、非連続型の Switch の各値(State)に対するストリングのオフセットの種類を指定します。これは、State オブジェクトの XmNxrtGaOffsetUseDefault が True の時にのみ適用されます。

(その場合、ストリングのオフセットは、親の Circular Dial ウィジェットによってコントロールされます。)

```
XRTGA_OFFSET_TYPE_AUTOMATIC
XRTGA_OFFSET_TYPE_CIRCULAR
XRTGA_OFFSET_TYPE_HORIZONTAL
XRTGA_OFFSET_TYPE_VERTICAL
```

XrtGaReason

Reason 構造体メンバとして、ほとんどのコールバックに使用されます。

```
XRTGA_REASON_VALUE_CHANGE_BEGIN = 100
XRTGA_REASON_VALUE_CHANGE_END
XRTGA_REASON_INTERACTION_START
XRTGA_REASON_INTERACTION_MOVE
XRTGA_REASON_INTERACTION_END
XRTGA_REASON_RESET_COUNTER_BEGIN
XRTGA_REASON_RESET_COUNTER_END
XRTGA_REASON_MAP
XRTGA_REASON_EXPOSE
XRTGA_REASON_RESIZE
XRTGA_REASON_ZOOM
XRTGA_REASON_TRANSLATE
```

XrtGaShape

Linear Dial ウィジェット、Frame、Icon、Range オブジェクトの XmNxrtGaShape で使用され、表示される形を指定します。

Linear Dial ウィジェットの有効な値は、RECTANGLE、CIRCLE、THERMOMETER、TANK です。

Frame オブジェクトの有効な値は RECTANGLE、CIRCLE、IMAGE です。

Icon オブジェクトの有効な値は、オンとオフの色が使用される RECTANGLE、CIRCLE または、指定されたイメージが使用される IMAGE です。Icon のサイズは親によって決定されます。

Range オブジェクトでは、セットされる Scale が Circular の場合は環状の形態が使用されるので、このリソースは無視されます。Scale がそれ以外の場合は、下記の値が有効です。

```
XRTGA_SHAPE_RECTANGLE
XRTGA_SHAPE_CIRCLE
XRTGA_SHAPE_THERMOMETER
XRTGA_SHAPE_TANK
XRTGA_SHAPE_IMAGE
```

XrtGaSizingPolicy

LEDIcon ウィジェットの XmNxrtGaSizingPolicy で使用され、ウィジェットのサイズに対する Icon のサイズをコントロールします。有効な値は WIDGET、ICON、BY_POSITION、CENTER です。

WIDGET: ウィジェットの幅と高さがアイコンのサイズを決定し、ユーザがアイコンのサイズを変更することはできません。

ICON: アイコンの幅と高さがウィジェットのサイズを決定し、ユーザがウィジェットのサイズを変更することはできません。

BY_POSITION: XmNxrtGaXOrigin と XmNxrtGaYOrigin が LEDIcon のグループの位置を決定し、XmNxrtGaIconHeight と XmNxrtGaIconWidth がアイコンのサイズを決定します。全てのアイコンの表示に十分なスペースがウィジェットにない場合、表示しきれない部分は切り取られます。

CENTER: XmNxrtGaIconHeight と XmNxrtGaIconWidth がアイコンのサイズを決定し、LEDIcon のグループはウィジェットの中央に配置されます。全てのアイコンの表示に十分なスペースがウィジェットにない場合、表示しきれない部分は切り取られます。

```
XRTGA_SIZING_POLICY_WIDGET
XRTGA_SIZING_POLICY_ICON
XRTGA_SIZING_POLICY_BY_POSITION
XRTGA_SIZING_POLICY_CENTER
```

XrtGaStringMode

Text オブジェクトの XmNxrtGaStringMode に使用される列挙法です。STRING が選択されている場合、通常のストリングが使用され、XmNxrtGaString リソースによって指されます。XM_STRING が選択されている場合は、XmString が使用され、XmNxrtGaXmString リソースによって指されます。

```
XRTGA_STRING_MODE_STRING
XRTGA_STRING_MODE_XM_STRING
```

XrtGaSwitchType

Switch オブジェクトの XmNxrtGaSwitchType に使用され、Switch の種類を指定します。

```
XRTGA_SWITCH_TYPE_CIRCLE
XRTGA_SWITCH_TYPE_POINTER
XRTGA_SWITCH_TYPE_ARROW
XRTGA_SWITCH_TYPE_FAUCET
XRTGA_SWITCH_TYPE_IMAGE
```

XrtGaTickType

Tick オブジェクトの XmNxrtGaTickType に使用され、Tick の種類を指定します。

```
XRTGA_TICK_TYPE_RECTANGLE
XRTGA_TICK_TYPE_CIRCLE
XRTGA_TICK_TYPE_DIAMOND
XRTGA_TICK_TYPE_TRIANGLE
```

XrtGaValueType

LEDNumerical ウィジェットの XmNxrtGaValueType に使用され、表示される値の種類を指定します。NUMERIC が選択されている場合、値は XmNxrtGaValue から取得されます。ALPHANUMERIC が選択されている場合は、値は XmNxrtGaText から取得されます。

```
XRTGA_VALUE_TYPE_NUMERIC
XRTGA_VALUE_TYPE_ALPHANUMERIC
```

B.2 構造体

XrtGaCallbackStruct

全ての XRT/gauge に使用され、Expose と Resize のコールバックに渡される情報を定義します。

```
typedef struct {
 int     reason; /* read only */
 XEvent  *event; /* read only */
 Window  window; /* read only */
 Region  region; /* read only */
 int     change; /* read only */
} XrtGaCallbackStruct;
```

XrtGaDrawData

Indicator、Marker、Needle、Switch オブジェクトが、user_data として XrtGaDrawFunc に情報を渡すときに使用されます。

```
typedef struct {
 Position    x;      /* X position of drawing origin */
 Position    y;      /* Y position of drawing origin */
 Dimension   width;  /* Width of draw object */
 Dimension   height; /* Height of draw object */
 double      angle;  /* Angle at which
                        the object is to be drawn */
 Position    x_offset;   /* x distance from (x, y) */
 Position    y_offset;   /* y distance from (x, y) */
} XrtGaDrawData;
```

XrtGaInteractionCallbackStruct

オブジェクトの XmNxrtGaInteractionCallback に渡されます。

```
typedef struct {
 int     reason;     /* read only */
 XEvent  *event;     /* read only */
 Boolean doit;       /* read/write when reason is not
                        XRTGA_REASON_INTERACTION_END. */
} XrtGaInteractionCallbackStruct;
```

XrtGaMapCallbackStruct

Circular Scale と Linear Scale オブジェクトに使用され、オブジェクトの XmNxrtGaMapCallback に渡されます。

```
typedef struct {
 int             reason;     /* read only */
 XEvent          *event;     /* read only */
 String          *params;    /* read only */
 Cardinal        nparams;    /* read only */
 XrtGaObject     scale;      /* read only */
 XrtGaMapResult  map;        /* read only */
} XrtGaMapCallbackStruct;
```

XrtGaMapResult

Circular Scale と Linear Scale オブジェクトに使用され、*XrtGaMap()*が呼び出されるとセットされます。

```
typedef struct {
 Position    pix_x; /* X pixel position */
 Position    pix_y; /* Y pixel position */
 double      value; /* Value of (pix_x, pix_y) in scale */
 double      extent; /* Extent along radius, width, height */
 double      angle; /* Angle for circular scales */
 Position    x_offset; /* X distance from scale origin */
 Position    y_offset; /* Y distance from scale origin */
} XrtGaMapResult;
```

XrtGaPolyDrawStruct

Switch と Tick オブジェクトに使用され、*XrtGaDrawPolyTransform()*に渡されます。

```
typedef struct {
 int     operations; /* Bitwise OR of possible XRTGA_* ops */
 Position    x; /* X position to translate to. */
 Position    y; /* Y position to translate to. */
 Dimension   width; /* Width to scale to. */
 Dimension   height; /* Height to scale to. */
 Double      angle; /* Angle to rotate to (0 degrees is East
                 and angles increase counterclockwise.)*/
 Xpoint      *shape; /* List of points */
 Int         npoints; /* Number of points in "shape". */
} XrtGaPolyDrawStruct;
```

XrtGaTransformCallbackStruct

Circular Scale と Linear Scale オブジェクトに使用され、オブジェクトの XmNxrtGaTransformCallback に渡されます。パラメータが-1 の時は、値が変更されておらず、これからも変更されないことを意味します。

```
typedef struct {
 int     reason; /* read only, #XrtGaReason */
 XEvent  *event; /* read only */
 String  *params; /* read only */
 Cardinal    nparams; /* read only */
 Dimension   new_size; /* New radius size for circular
                 scales */
 Dimension   new_height; /* New height for linear scales */
 Dimension   new_width; /* New width for linear scales */
 Position    new_x; /* New X location for both linear and
                 circular scales */
 Position    new_y; /* New Y location for both linear and
                 circular scales */
 Boolean     doit; /* Default: True. User sets to False
                 to disallow the operation */
} XrtGaTransformCallbackStruct;
```

XrtGaUserDrawInfo

*XrtGaGetUserDrawInfo()*から返される構造体です。

```
typedef struct {
 Display *display; /* Display of this widget/object */
 int     screen; /* Screen number */
 Window  window; /* Window ID */
 Pixmap  pixmap; /* Pixmap used for double buffering
                 (may be \c XmUNSPECIFIED_PIXMAP). */
 Colormap    cmap; /* Colormap assigned to this
                      widget or object */
 VisualID    vid; /* Visual ID to be used. */
 Int         visual_class; /* VisualClass (PseudoColor,
                      TrueColor, and so on.). */
 Dimension   width; /* Widget window's width */
 Dimension   height; /* Widget window's height */
 Int         depth; /* Widget window's depth */
 Boolean     double_buffer; /* XmNxrtGaDoubleBuffer */
} XrtGaUserDrawInfo;
```

XrtGaValueChangedCallbackStruct

LEDIcon と LEDNumerical ウィジェット、Counter、Indicator、Item、Needle、Switch オブジェクトに使用され、ウィジェットまたはオブジェクトの XmNxrtGaValueChangedCallback に渡されます。

```
typedef struct {
 int     reason; /* read only */
 XEvent  *event; /* read only (will be NULL for
                 programatic changes) */
 double  value; /* read/write when reason is
                 #XRTGA_REASON_VALUE_CHANGE_BEGIN. */
 Boolean doit; /* read/write when reason is
                 #XRTGA_REASON_VALUE_CHANGE_BEGIN. */
} XrtGaValueChangedCallbackStruct;
```

C

リソースの型とクラス

C.1 Center オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaCircularScale	XmCXrtGaCircularScale	XmRXrtGaCircularScale
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaRadius	XmCXrtGaRadius	XmRXrtGaDouble

C.2 Circular Dial ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaCircularScale	XmCXrtGaCircularScale	XmRXrtGaCircularScale
XmNxrtGaCounterList	XmCXrtGaCounterList	XmRXrtGaCounterList
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaExposeCallback	XmCCallback	XmRCallback
XmNxrtGaFrame	XmCXrtGaFrame	XmRXrtGaFrame

Name	Class	Type
XmNxrtGaHighlightColor	XmCXrtGaHighlightColor	XmRPixel
XmNxrtGaHighlightSelectedState	XmCXrtGaHighlightSelectedState	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaInteractionCallback	XmCCallback	XmRCallback
XmNxrtGaLineInnerRadius	XmCXrtGaLineRadius	XmRXrtGaDouble
XmNxrtGaLineOuterRadius	XmCXrtGaLineRadius	XmRXrtGaDouble
XmNxrtGaMapCallback	XmCCallback	XmRCallback
XmNxrtGaOffsetDistance	XmCXrtGaOffsetDistance	XmRXrtGaDouble
XmNxrtGaOffsetType	XmCXrtGaOffsetType	XmRXrtGaOffsetType
XmNxrtGaMarker	XmCXrtGaMarker	XmRXrtGaMarker
XmNxrtGaRangeList	XmCXrtGaRangeList	XmRXrtGaRangeList
XmNxrtGaRenderList	XmCXrtGaRenderList	XmRXrtGaRenderList
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaStateList	XmCXrtGaStateList	XmRXrtGaStateList
XmNxrtGaSwitch	XmCXrtGaSwitch	XmRXrtGaSwitch
XmNxrtGaTextList	XmCXrtGaTextList	XmRXrtGaTextList
XmNxrtGaTickList	XmCXrtGaTickList	XmRXrtGaTickList
XmNxrtGaUseRenderList	XmCXrtGaUseRenderList	XmRBoolean

C.3 Circular Gauge ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaCenterList	XmCXrtGaCenterList	XmRXrtGaCenterList
XmNxrtGaCircularScaleList	XmCXrtGaCircularScaleList	XmRXrtGaCircularScaleList
XmNxrtGaCounterList	XmCXrtGaCounterList	XmRXrtGaCounterList
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaExposeCallback	XmCCallback	XmRCallback
XmNxrtGaFrame	XmCXrtGaFrame	XmRXrtGaFrame

Name	Class	Type
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaInteractionCallback	XmCCallback	XmRCallback
XmNxrtGaMapCallback	XmCCallback	XmRCallback
XmNxrtGaNeedleList	XmCXrtGaNeedleList	XmRXrtGaNeedleList
XmNxrtGaRangeList	XmCXrtGaRangeList	XmRXrtGaRangeList
XmNxrtGaRenderList	XmCXrtGaRenderList	XmRXrtGaRenderList
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaTextList	XmCXrtGaTextList	XmRXrtGaTextList
XmNxrtGaTickList	XmCXrtGaTickList	XmRXrtGaTickList
XmNxrtGaUseRenderList	XmCXrtGaUseRenderList	XmRBoolean

C.4 Circular Scale オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaAllowTransform	XmCXrtGaAllowTransform	XmRBoolean
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDirection	XmCXrtGaDirection	XmRXrtGaDirection
XmNxrtGaMax	XmCXrtGaLimits	XmRXrtGaDouble
XmNxrtGaMin	XmCXrtGaLimits	XmRXrtGaDouble
XmNxrtGaRadiusSize	XmCXrtGaRadiusSize	XmRDimension
XmNxrtGaStartAngle	XmCXrtGaAngle	XmRXrtGaDouble
XmNxrtGaStopAngle	XmCXrtGaAngle	XmRXrtGaDouble
XmNxrtGaTransformCallback	XmCCallback	XmRCallback
XmNxrtGaXOrigin	XmCXrtGaPosition	XmRPosition
XmNxrtGaYOrigin	XmCXrtGaPosition	XmRPosition

C.5 Counter オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNfontList	XmCFontList	XmRFontList
XmNheight	XmCHeight	XmRDimension
XmNwidth	XmCWidth	XmRDimension
XmNx	XmCX	XmRPosition
XmNy	XmCY	XmRPosition
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaBorderWidth	XmCXrtGaDimension	XmRDimension
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDecimalPlaces	XmCXrtGaDecimalPlaces	XmRInt
XmNxrtGaDigitBackgroundColor	XmCXrtGaColor	XmRPixel
XmNxrtGaDigitForegroundColor	XmCXrtGaColor	XmRPixel
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaNumDigits	XmCXrtGaNumDigits	XmRInt
XmNxrtGaResetAlignment	XmCXrtGaResetAlignment	XmRXrtGaAlignment
XmNxrtGaResetBackgroundColor	XmCBackground	XmRPixel
XmNxrtGaResetHeight	XmCHeight	XmRDimension
XmNxrtGaResetImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaResetOffset	XmCOffset	XmRDimension
XmNxrtGaResetShape	XmCXrtGaShape	XmRXrtGaShape
XmNxrtGaResetShow	XmCXrtGaResetShow	XmRBoolean
XmNxrtGaResetWidth	XmCWidth	XmRDimension
XmNxrtGaSpacingHorizontal	XmCXrtGaSpacing	XmRDimension
XmNxrtGaSpacingVertical	XmCXrtGaSpacing	XmRDimension
XmNxrtGaTransitionRate	XmCXrtGaTransition	XmRInt
XmNxrtGaTransitionSteps	XmCXrtGaTransition	XmRInt
XmNxrtGaUseTransition	XmCXrtGaUseTransition	XmRBoolean
XmNxrtGaValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback

C.6 Frame オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaBottomMargin	XmCXrtGaMargin	XmRInt
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaLeftMargin	XmCXrtGaMargin	XmRInt
XmNxrtGaRightMargin	XmCXrtGaMargin	XmRInt
XmNxrtGaShape	XmCXrtGaShape	XmRXrtGaShape
XmNxrtGaTopMargin	XmCXrtGaMargin	XmRInt

C.7 Icon オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaOffColor	XmCXrtGaColor	XmRPixel
XmNxrtGaOffImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaOnColor	XmCXrtGaColor	XmRPixel
XmNxrtGaOnImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaShape	XmCXrtGaShape	XmRXrtGaShape

C.8 Indicator オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNwidth	XmCWidth	XmRDimension
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDrawFunc	XmCXrtGaDrawFunc	XmRFunction
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage

Name	Class	Type
XmNxrtGaImageRotate	XmCXrtGaImageRotate	XmRBoolean
XmNxrtGaIndicatorType	XmCXrtGaIndicatorType	XmRXrtGaIndicatorType
XmNxrtGaInnerExtent	XmCXrtGaExtent	XmRXrtGaDouble
XmNxrtGaInteractionType	XmCXrtGaInteractionType	XmRXrtGaInteractionType
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaLinearScale	XmCXrtGaLinearScale	XmRXrtGaLinearScale
XmNxrtGaOuterExtent	XmCXrtGaExtent	XmRXrtGaDouble
XmNxrtGaValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback

C.9 Item オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaLabelExtent	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback

C.10 LEDIcon ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDecayRate	XmCXrtGaDecayRate	XmRInt
XmNxrtGaDirection	XmCXrtGaDirection	XmRXrtGaDirection
XmNxrtGaDisplayMode	XmCXrtGaDisplayMode	XmRXrtGaDisplayMode
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaExposeCallback	XmCCallback	XmRCallback

Name	Class	Type
XmNxrtGaIconHeight	XmCXrtGaDimension	XmRDimension
XmNxrtGaIconList	XmCXrtGaIconList	XmRXrtGaIconList
XmNxrtGaIconSpacing	XmCXrtGaDimension	XmRDimension
XmNxrtGaIconWidth	XmCXrtGaDimension	XmRDimension
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaInteractionCallback	XmCCallback	XmRCallback
XmNxrtGaInteractionType	XmCXrtGaInteractionType	XmRXrtGaInteractionType
XmNxrtGaLEDStartValue	XmCXrtGaIntValue	XmRInt
XmNxrtGaLEDValue	XmCXrtGaIntValue	XmRInt
XmNxrtGaMaxIcon	XmCXrtGaMaxIcon	XmRBoolean
XmNxrtGaOrientation	XmCXrtGaOrientation	XmROrientation
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaSizingPolicy	XmCXrtGaSizingPolicy	XmRXrtGaSizingPolicy
XmNxrtGaSpacingHorizontal	XmCXrtGaSpacing	XmRDimension
XmNxrtGaSpacingVertical	XmCXrtGaSpacing	XmRDimension
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback
XmNxrtGaXOrigin	XmCXrtGaPosition	XmRPosition
XmNxrtGaYOrigin	XmCXrtGaPosition	XmRPosition

C.11 LEDNumerical ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaColorMixPercentage	XmCXrtGaColorMixPercentage	XmRInt
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDecimalPlaces	XmCXrtGaDecimalPlaces	XmRInt
XmNxrtGaDecimalPosition	XmCXrtGaDecimalPosition	XmRPosition
XmNxrtGaDecimalSize	XmCXrtGaDecimalSize	XmRDimension
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaExposeCallback	XmCCallback	XmRCallback

Name	Class	Type
XmNxrtGaFixedDecimal	XmCXrtGaFixedDecimal	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaLeadingOne	XmCXrtGaLeadingOne	XmRBoolean
XmNxrtGaLeadingPlusMinus	XmCXrtGaLeadingPlusMinus	XmRBoolean
XmNxrtGaLeadingZeros	XmCXrtGaLeadingZeros	XmRBoolean
XmNxrtGaNumDigits	XmCXrtGaNumDigits	XmRInt
XmNxrtGaOffColor	XmCXrtGaColor	XmRPixel
XmNxrtGaOnColor	XmCXrtGaColor	XmRPixel
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaSegmentSpacing	XmCXrtGaSpacing	XmRDimension
XmNxrtGaSegmentWidth	XmCXrtGaDimension	XmRDimension
XmNxrtGaSlopeOffset	XmCXrtGaSlopeOffset	XmRPosition
XmNxrtGaSpacingHorizontal	XmCXrtGaSpacing	XmRDimension
XmNxrtGaSpacingVertical	XmCXrtGaSpacing	XmRDimension
XmNxrtGaText	XmCXrtGaText	XmRString
XmNxrtGaValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback
XmNxrtGaValueType	XmCXrtGaValueType	XmRXrtGaValueType

C.12 Linear Dial ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaDrawFunc	XmCXrtGaDrawFunc	XmRFunction
XmNxrtGaExposeCallback	XmCCallback	XmRCallback
XmNxrtGaFrame	XmCXrtGaFrame	XmRXrtGaFrame
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout

Name	Class	Type
XmNxrtGaIndicator	XmCXrtGaIndicator	XmRXrtGaIndicator
XmNxrtGaInteractionCallback	XmCCallback	XmRCallback
XmNxrtGaLinearScale	XmCXrtGaLinearScale	XmRXrtGaLinearScale
XmNxrtGaMapCallback	XmCCallback	XmRCallback
XmNxrtGaOffColor	XmCXrtGaColor	XmRPixel
XmNxrtGaOffImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaOnColor	XmCXrtGaColor	XmRPixel
XmNxrtGaOnImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaRenderList	XmCXrtGaRenderList	XmRXrtGaRenderList
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaShape	XmCXrtGaShape	XmRXrtGaShape
XmNxrtGaTextList	XmCXrtGaTextList	XmRXrtGaTextList
XmNxrtGaTickList	XmCXrtGaTickList	XmRXrtGaTickList
XmNxrtGaUseRenderList	XmCXrtGaUseRenderList	XmRBoolean

C.13 Linear Gauge ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaExposeCallback	XmCCallback	XmRCallback
XmNxrtGaFrame	XmCXrtGaFrame	XmRXrtGaFrame
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaIndicatorList	XmCXrtGaIndicatorList	XmRXrtGaIndicatorList
XmNxrtGaInteractionCallback	XmCCallback	XmRCallback
XmNxrtGaLinearScaleList	XmCXrtGaLinearScaleList	XmRXrtGaLinearScaleList
XmNxrtGaMapCallback	XmCCallback	XmRCallback
XmNxrtGaRangeList	XmCXrtGaRangeList	XmRXrtGaRangeList
XmNxrtGaRenderList	XmCXrtGaRenderList	XmRXrtGaRenderList

Name	Class	Type
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaTextList	XmCXrtGaTextList	XmRXrtGaTextList
XmNxrtGaTickList	XmCXrtGaTickList	XmRXrtGaTickList
XmNxrtGaUseRenderList	XmCXrtGaUseRenderList	XmRBoolean

C.14 Linear Scale オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNheight	XmCHeight	XmRDimension
XmNwidth	XmCWidth	XmRDimension
XmNx	XmCX	XmRPosition
XmNy	XmCY	XmRPosition
XmNxrtGaAllowTransform	XmCXrtGaAllowTransform	XmRBoolean
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDirection	XmCXrtGaDirection	XmRXrtGaDirection
XmNxrtGaMax	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaMin	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaOrientation	XmCXrtGaOrientation	XmROrientation
XmNxrtGaTransformCallback	XmCCallback	XmRCallback

C.15 Marker オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNwidth	XmCWidth	XmRDimension
XmNxrtGaCircularScale	XmCXrtGaCircularScale	XmRXrtGaCircularScale
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDrawFunc	XmCXrtGaDrawFunc	XmRFunction
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage

Name	Class	Type
XmNxrtGaImageRotate	XmCXrtGaImageRotate	XmRBoolean
XmNxrtGaInnerRadius	XmCXrtGaInnerRadius	XmRXrtGaDouble
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaMarkerType	XmCXrtGaMarkerType	XmRXrtGaMarkerType
XmNxrtGaOuterRadius	XmCXrtGaOuterRadius	XmRXrtGaDouble

C.16 Needle オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNwidth	XmCWidth	XmRDimension
XmNxrtGaCircularScale	XmCXrtGaCircularScale	XmRXrtGaCircularScale
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDrawFunc	XmCXrtGaDrawFunc	XmRFunction
XmNxrtGaInteractionType	XmCXrtGaInteractionType	XmRXrtGaInteractionType
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaLength	XmCXrtGaLength	XmRXrtGaDouble
XmNxrtGaNeedleType	XmCXrtGaNeedleType	XmRXrtGaNeedleType
XmNxrtGaValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback

C.17 Percent ウィジェット

Name	Class	Type
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaCalcMode	XmCXrtGaCalcMode	XmRXrtGaCalcMode
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDoubleBuffer	XmCXrtGaDoubleBuffer	XmRBoolean
XmNxrtGaExposeCallback	XmCCallback	XmRCallback
XmNxrtGaFrame	XmCXrtGaFrame	XmRXrtGaFrame
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage

Name	Class	Type
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaItemList	XmCXrtGaItemList	XmRXrtGaItemList
XmNxrtGaMapCallback	XmCCallback	XmRCallback
XmNxrtGaRenderList	XmCXrtGaRenderList	XmRXrtGaRenderList
XmNxrtGaRepaint	XmCXrtGaRepaint	XmRBoolean
XmNxrtGaResizeCallback	XmCCallback	XmRCallback
XmNxrtGaScale	XmCXrtGaScale	XmRXrtGaScale
XmNxrtGaTextList	XmCXrtGaTextList	XmRXrtGaTextList
XmNxrtGaUseRenderList	XmCXrtGaUseRenderList	XmRBoolean

C.18 Range オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaInnerExtent	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaOffset	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaOuterExtent	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaScale	XmCXrtGaScale	XmRXrtGaScale
XmNxrtGaShape	XmCXrtGaShape	XmRXrtGaShape
XmNxrtGaStartValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaStopValue	XmCXrtGaDoubleValue	XmRXrtGaDouble

C.19 State オブジェクト

Name	Class	Type
XmNfontList	XmCFontList	XmRFontList
XmNforeground	XmCForeground	XmRPixel

Name	Class	Type
XmNxrtGaAlignment	XmCXrtGaAlignment	XmRXrtGaAlignment
XmNxrtGaAlignmentUseDefault	XmCXrtGaAlignmentUseDefault	XmRBoolean
XmNxrtGaAngle	XmCXrtGaAngle	XmRXrtGaDouble
XmNxrtGaAngleUseDefault	XmCXrtGaAngleUseDefault	XmRBoolean
XmNxrtGaCircularScale	XmCXrtGaCircularScale	XmRXrtGaCircularScale
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaLineDisplay	XmCXrtGaLineDisplay	XmRBoolean
XmNxrtGaLineInnerRadius	XmCXrtGaLineRadius	XmRXrtGaDouble
XmNxrtGaLineInnerRadiusUseDefault	XmCXrtGaLineRadiusUseDefault	XmRBoolean
XmNxrtGaLineOuterRadius	XmCXrtGaLineRadius	XmRXrtGaDouble
XmNxrtGaLineOuterRadiusUseDefault	XmCXrtGaLineRadiusUseDefault	XmRBoolean
XmNxrtGaLinePattern	XmCXrtGaLinePattern	XmRXrtGaLinePattern
XmNxrtGaLineThickness	XmCXrtGaLineThickness	XmRInt
XmNxrtGaOffsetUseDefault	XmCXrtGaOffsetUseDefault	XmRBoolean
XmNxrtGaOffsetX	XmCXrtGaOffsetX	XmRPosition
XmNxrtGaOffsetY	XmCXrtGaOffsetY	XmRPosition
XmNxrtGaText	XmCXrtGaText	XmRString

C.20 Switch オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNxrtGaCircularScale	XmCXrtGaCircularScale	XmRXrtGaCircularScale
XmNxrtGaContinuous	XmCXrtGaContinuous	XmRBoolean
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDrawFunc	XmCXrtGaDrawFunc	XmRFunction
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageRotate	XmCXrtGaImageRotate	XmRBoolean
XmNxrtGaIncrementValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaInteractionType	XmCXrtGaInteractionType	XmRXrtGaInteractionType
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean

Name	Class	Type
XmNxrtGaMultipleTurnEnabled	XmCXrtGaMultipleTurnEnabled	XmRBoolean
XmNxrtGaMultipleTurnValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaRadius	XmCXrtGaRadius	XmRXrtGaDouble
XmNxrtGaSwitchType	XmCXrtGaSwitchType	XmRXrtGaSwitchType
XmNxrtGaValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaValueChangedCallback	XmCCallback	XmRCallback

C.21 Text オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNfontList	XmCFontList	XmRFontList
XmNforeground	XmCForeground	XmRPixel
XmNheight	XmCHeight	XmRDimension
XmNwidth	XmCWidth	XmRDimension
XmNx	XmCX	XmRPosition
XmNy	XmCY	XmRPosition
XmNxrtGaAlignment	XmCXrtGaAlignment	XmRXrtGaAlignment
XmNxrtGaAllowWidgetPositioning	XmCXrtGaAllowWidgetPositioning	XmRBoolean
XmNxrtGaBorder	XmCXrtGaBorder	XmRXrtGaBorder
XmNxrtGaBorderWidth	XmCXrtGaDimension	XmRDimension
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaImage	XmCXrtGaImage	XmRXrtGaImage
XmNxrtGaImageLayout	XmCXrtGaImageLayout	XmRXrtGaImageLayout
XmNxrtGaImageMinimumSize	XmCXrtGaImageMinimumSize	XmRBoolean
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaString	XmCString	XmRString
XmNxrtGaStringMode	XmCXrtGaStringMode	XmRXrtGaStringMode
XmNxrtGaValueObject	XmCXrtGaValueObject	XmRXrtGaValueObject
XmNxrtGaXmString	XmCXmString	XmRXmString

C.22 Tick オブジェクト

Name	Class	Type
XmNbackground	XmCBackground	XmRPixel
XmNfontList	XmCFontList	XmRFontList
XmNwidth	XmCWidth	XmRDimension
XmNxrtGaAutomatic	XmCXrtGaAutomatic	XmRBoolean
XmNxrtGaDebug	XmCXrtGaDebug	XmRBoolean
XmNxrtGaDrawLabels	XmCXrtGaDrawLabels	XmRBoolean
XmNxrtGaIncrementValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaInnerExtent	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaIsShowing	XmCXrtGaIsShowing	XmRBoolean
XmNxrtGaLabelExtent	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaLabelFunc	XmCXrtGaFunction	XmRFunction
XmNxrtGaOffset	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaOuterExtent	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaPrecision	XmCXrtGaPrecision	XmRInt
XmNxrtGaPrecisionUseDefault	XmCXrtGaPrecisionUseDefault	XmRBoolean
XmNxrtGaScale	XmCXrtGaScale	XmRXrtGaScale
XmNxrtGaStartValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaStopValue	XmCXrtGaDoubleValue	XmRXrtGaDouble
XmNxrtGaTickType	XmCXrtGaTickType	XmRXrtGaTickType

D

継承されたリソース

XRT/gauge の各ウィジェットが使用する継承されたリソースの一覧です。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。

XmPrimitive リソース･セット

Name	Type	Default	Access
XmNbottomShadowColor	Pixel	dynamic	CSG
XmNbottomShadowPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNforeground	Pixel	dynamic	CSG
XmNhelpCallback	XtCallbackList	NULL	C
XmNhighlightColor	Pixel	dynamic	CSG
XmNhighlightOnEnter	Boolean	False	CSG
XmNhighlightPixmap	Pixmap	dynamic	CSG
XmNhighlightThickness	Dimension	2	CSG
XmNnavigationType	XmNavigationType	XmNONE	G
XmNshadowThickness	Dimension	2	CSG
XmNtopShadowColor	Pixel	dynamic	CSG
XmNtopShadowPixmap	Pixmap	dynamic	CSG
XmNtraversalOn	Boolean	True	CSG
XmNunitType	unsigned char	dynamic	CSG
XmNuserData	Pointer	NULL	CSG

Core リソース・セット

Name	Type	Default	Access
XmNaccelerators	XtAccelerators	NULL	CSG
XmNancestorSensitive	Boolean	dynamic	G
XmNbackground	Pixel	dynamic	CSG
XmNbackgroundPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNborderColor	Pixel	XtDefaultForeground	CSG
XmNborderPixmap	Pixmap	XmUNSPECIFIED_PIXMAP	CSG
XmNborderWidth	Dimension	0	CSG
XmNcolormap	Colormap	dynamic	CG
XmNdepth	int	dynamic	CG
XmNdestroyCallback	XtCallbackList	NULL	C
XmNheight	Dimension	dynamic	CSG
XmNinitialResourcesPersistent	Boolean	True	C
XmNmappedWhenManaged	Boolean	True	CSG
XmNscreen	Screen *	dynamic	CG
XmNsensitive	Boolean	True	CSG
XmNtranslations	XtTranslations	NULL	CSG
XmNwidth	Dimension	dynamic	CSG
XmNx	Position	0	CSG
XmNy	Position	0	CSG

E

リソース・ファイルと GUI ビルダ

GUIビルダ・ツールを使用してXRT/gaugeのリソースをリソース・ファイルでセットする場合の構文の一覧です。この章の最後には、サンプルのプログラムとリソース・ファイルが提供されています。XRT/gauge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とオブジェクト・リスト・リソースを除く全てのXRT/gaugeリソースは、全てリソース・ファイルでセットできます。

E.1 構文

コメント

(!)の後のテキストは無視されます。

継続

行の一番最後をバック・スラッシュ(\)にすると、次の行に継続します。

Boolean

大文字、小文字、大・小混合文字の"True"または"False"。

列挙型

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XRTGA_と残りのアンダースコアがありません。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XRTGA_BORDER_SHADOWがCircular Dialウィジェットの XmNxrtGaBorderに使用されるようにしています。

```
*.XmXrtGaCircularDial.xrtGaBorder: BorderShadow
*.XmXrtGaCircularDial.xrtGaBorder: BORDERSHADOW
*.XmXrtGaCircularDial.xrtGaBorder: bordershadow
```

イメージ

XmRXrtGaImage 型のリソースは有効なファイル名で指定します。ファイル名がチェックされ、デフォルトのイメージ型かユーザによって登録された型に変換されます。

例

```
*.XmXrtGaLinearDial.xrtGaOnImage: $XRTHOME/src/icons/smile32.xpm
```

オブジェクト

XmRXrtGaObject 型のリソースは、リソース・ファイルで指定できます。リソース・ファイルで指定されたオブジェクトは必ず XtName で取得されます。また、参照されているオブジェクトは、それを参照しているオブジェクトまたはウィジェットの前に作成されていなければなりません。例えば、Range オブジェクトが XmNxrtGaScale をリソース・ファイルでセットしようとする場合、Range オブジェクトは Scale の後に作成される必要があります。この章の最後にあるサンプル・コードでは、オブジェクトの作成の正しい順序が示されていますので参考にしてください。

オブジェクトのリスト

XmRXrtGaObjectList 型のリソースは、リソース・ファイルでは指定できません。

ストリング値

ストリングには、標準の Motif ストリングの指定法を使用することができます。

E.2 サンプル・コードとリソース・ファイル

図47は、この章にあるサンプル・コードとリソース・ファイルによって定義されています。

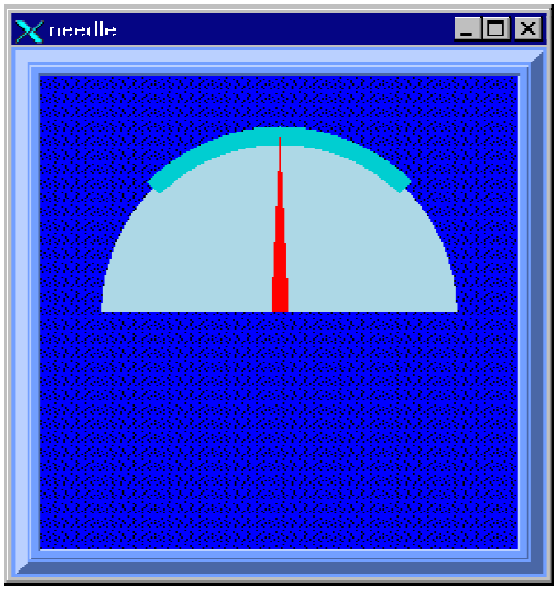


図47　リソースがリソース・ファイルでセットされているCircular Gauge

E.2.1 サンプル・プログラム

Circular Gaugeウィジェットを作成し、Formに配置する基本的なプログラムです。

```
#include <Xm/Xm.h>
#include <Xm/Form.h>
#include <Xm/XrtGaugeAll.h>
#include "mtgauge.h"

int
main(argc, argv)
int argc;
char *argv[];
{
 XtAppContext     app_context;
 Widget           toplevel, form, cgauge, scale, range, needle;

 putenv("XENVIRONMENT=./needle.res");

 /* Initialize Intrinsics */
 toplevel = XtVaAppInitialize(&app_context, "CGauge", NULL, 0,
                  &argc, argv, NULL, NULL);
 /* Create Form*/
 form = XtVaCreateManagedWidget("form",
                  xmFormWidgetClass, toplevel,
                  NULL);
 /* Create Gauge */
 cgauge =
```

```
        XtVaCreateManagedWidget("circular_gauge",
                    xmXrtGaCircularGaugeWidgetClass, form,
                    NULL);
    /* Create Basic Scale */
    scale = XtVaCreateWidget("scale",
                    xmXrtGaCircularScaleObjectClass, cgauge,
                    NULL);
    /* Create Range */
    range = XtVaCreateWidget("range",
                    xmXrtGaRangeObjectClass, cgauge,
                    NULL);
    /* Create Needle */
    needle = XtVaCreateWidget("needle",
                    xmXrtGaNeedleObjectClass, cgauge,
                    NULL);
    XtRealizeWidget(toplevel);
    XtAppMainLoop(app_context);
}
```

これらのリソースは、リソース・ファイルでセットされています。作成されていないオブジェクトの参照は許可されていませんので、Range や Needle の前に Scale が作成されていなければなりません。

```
CGauge*.circular_gauge.xrtGaImage:     $XRTHOME/src/icons/shading.xpm
CGauge*.circular_gauge.xrtGaImageLayout:    ImageLayoutTiled
CGauge*.circular_gauge.xrtGaBorder:         BorderFrameOut
CGauge*.circular_gauge.shadowThickness:     14
CGauge*.scale.background:                   lightblue
CGauge*.scale.xrtGaStartAngle:              0.0
CGauge*.scale.xrtGaStopAngle:               180.0
CGauge*.scale.xrtGaMin:                     50.0
CGauge*.scale.xrtGaMax:                     150.0
CGauge*.scale.xrtGaDirection:               DirectionBackward
CGauge*.range.xrtGaScale:                   scale
CGauge*.range.xrtGaInnerExtent:             95%
CGauge*.range.xrtGaOuterExtent:             105%
CGauge*.range.xrtGaStartValue:              75.0
CGauge*.range.xrtGaStopValue:               125.0
CGauge*.range.background:                   DarkTurquoise
CGauge*.needle.background:                  red
CGauge*.needle.width:                       10
CGauge*.needle.xrtGaCircularScale:          scale
CGauge*.needle.xrtGaInteractionType:   InteractionTypeSnapToMouse
CGauge*.needle.xrtGaNeedleType:             NeedleTypeTriang
```

このプログラムをコンパイル、実行すると、図 47 のイメージが表示されます。

F

UIL 引数

XRT/gauge では、OSF の User Interface Language(UIL)を使用してウィジェットを作成することができます。UIL は、標準の Motif ウィジェットで使用されているデータ型しかサポートしませんので、XRT/gauge 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、`MrmNcreateCallbck` を使用してアプリケーションでセットすることができます。

この章では、XRT/gauge の各ウィジェットとオブジェクトの UIL 引数とそのデータ型及び reason がアルファベット順にリストアップされています。

UIL を使用したプログラムの例は、*$XRTHOME/src/gauge/examples/simple_cpp.c (using simple_cpp.uil)*にあります。

F.1 Center オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaCircularScale	widget
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaIsShowing	boolean
XmNxrtGaRadius	float

F.2 Circular Dial ウィジェットのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaFrame	widget
XmNxrtGaHighlightColor	color
XmNxrtGaHighlightSelectedState	boolean
XmNxrtGaImageLayout	integer
XmNxrtGaLineInnerRadius	float
XmNxrtGaLineOuterRadius	float
XmNxrtGaOffsetDistance	float
XmNxrtGaOffsetType	integer
XmNxrtGaMarker	widget
XmNxrtGaRepaint	boolean
XmNxrtGaSwitch	widget
XmNxrtGaUseRenderList	boolean

Reasons
XmNxrtGaInteractionCallback
XmNxrtGaExposeCallback
XmNxrtGaResizeCallback

F.3 Circular Gauge ウィジェットのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaDebug	boolean
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaFrame	widget
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaLineInnerRadius	float

UIL Argument Name	Argument Type
XmNxrtGaLineOuterRadius	float
XmNxrtGaRepaint	boolean
XmNxrtGaUseRenderList	boolean

Reasons
XmNxrtGaExposeCallback
XmNxrtGaInteractionCallback
XmNxrtGaResizeCallback

F.4 Circular Scale オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaAllowTransform	boolean
XmNxrtGaDebug	boolean
XmNxrtGaDirection	integer
XmNxrtGaMax	float
XmNxrtGaMin	float
XmNxrtGaRadiusSize	integer
XmNxrtGaStartAngle	float
XmNxrtGaStopAngle	float
XmNxrtGaXOrigin	integer
XmNxrtGaYOrigin	integer

Reasons
XmNxrtGaTransformCallback

F.5 Counter オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaBorderWidth	integer
XmNxrtGaDebug	boolean
XmNxrtGaDecimalPlaces	integer
XmNxrtGaDigitBackgroundColor	color
XmNxrtGaDigitForegroundColor	color
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaIsShowing	boolean
XmNxrtGaNumDigits	integer
XmNxrtGaResetAlignment	integer
XmNxrtGaResetHeight	integer
XmNxrtGaResetIcon	widget
XmNxrtGaResetOffset	integer
XmNxrtGaResetShow	boolean
XmNxrtGaResetWidth	integer
XmNxrtGaSpacingHorizontal	integer
XmNxrtGaSpacingVertical	integer
XmNxrtGaTransitionRate	integer
XmNxrtGaTransitionSteps	integer
XmNxrtGaUseTransition	boolean
XmNxrtGaValue	float

Reasons
XmNxrtGaValueChangedCallback

F.6 Frame オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaBottomMargin	integer
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaIsShowing	boolean
XmNxrtGaLeftMargin	integer
XmNxrtGaRightMargin	integer
XmNxrtGaShape	integer
XmNxrtGaTopMargin	integer

F.7 Icon オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaDebug	boolean
XmNxrtGaOffColor	color
XmNxrtGaOffImage	pixmap
XmNxrtGaOnColor	color
XmNxrtGaOnImage	pixmap
XmNxrtGaShape	integer

F.8 Indicator オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageRotate	boolean
XmNxrtGaIndicatorType	integer
XmNxrtGaInnerExtent	float
XmNxrtGaInteractionType	integer
XmNxrtGaIsShowing	boolean

UIL Argument Name	Argument Type
XmNxrtGaLinearScale	widget
XmNxrtGaOuterExtent	float
XmNxrtGaValue	float

Reasons
XmNxrtGaValueChangedCallback

F.9 Item オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaIsShowing	boolean
XmNxrtGaLabelExtent	float
XmNxrtGaValue	float

Reasons
XmNxrtGaValueChangedCallback

F.10 LEDIcon ウィジェットのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaDebug	boolean
XmNxrtGaDecayRate	integer
XmNxrtGaDirection	integer
XmNxrtGaDisplayMode	integer
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaIconHeight	integer
XmNxrtGaIconSpacing	integer
XmNxrtGaIconWidth	integer
XmNxrtGaImage	pixmap

UIL Argument Name	Argument Type
XmNxrtGaImageLayout	integer
XmNxrtGaInteractionType	integer
XmNxrtGaLEDStartValue	integer
XmNxrtGaLEDValue	integer
XmNxrtGaMaxIcon	boolean
XmNxrtGaOrientation	integer
XmNxrtGaRepaint	boolean
XmNxrtGaSizingPolicy	integer
XmNxrtGaSpacingHorizontal	integer
XmNxrtGaSpacingVertical	integer
XmNxrtGaXOrigin	integer
XmNxrtGaYOrigin	integer

Reasons
XmNxrtGaExposeCallback
XmNxrtGaInteractionCallback
XmNxrtGaResizeCallback
XmNxrtGaValueChangedCallback

F.11 LEDNumerical ウィジェットのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaColorMixPercentage	integer
XmNxrtGaDebug	boolean
XmNxrtGaDecimalPlaces	integer
XmNxrtGaDecimalPosition	integer
XmNxrtGaDecimalSize	integer
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaFixedDecimal	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaLeadingOne	boolean
XmNxrtGaLeadingPlusMinus	boolean
XmNxrtGaLeadingZeros	boolean
XmNxrtGaNumDigits	integer
XmNxrtGaOffColor	color
XmNxrtGaOnColor	color
XmNxrtGaRepaint	boolean
XmNxrtGaSegmentSpacing	integer
XmNxrtGaSegmentWidth	integer
XmNxrtGaSlopeOffset	integer
XmNxrtGaSpacingHorizontal	integer
XmNxrtGaSpacingVertical	integer
XmNxrtGaText	string
XmNxrtGaValue	float
XmNxrtGaValueType	integer

Reasons
XmNxrtGaExposeCallback
XmNxrtGaResizeCallback
XmNxrtGaValueChangedCallback

F.12 Linear Dial ウィジェットのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaDebug	boolean
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaFrame	widget
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaIndicator	widget
XmNxrtGaLinearScale	widget
XmNxrtGaOffColor	color
XmNxrtGaOffImage	pixmap
XmNxrtGaOnColor	color
XmNxrtGaOnImage	pixmap
XmNxrtGaRepaint	boolean
XmNxrtGaShape	integer
XmNxrtGaUseRenderList	boolean

Reasons
XmNxrtGaExposeCallback
XmNxrtGaInteractionCallback
XmNxrtGaResizeCallback

F.13 Linear Gauge ウィジェットのリソース

UILArgument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaDebug	boolean
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaFrame	widget
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer

UILArgument Name	Argument Type
XmNxrtGaRepaint	Boolean
XmNxrtGaUseRenderList	Boolean

Reasons
XmNxrtGaExposeCallback
XmNxrtGaInteractionCallback
XmNxrtGaResizeCallback

F.14 Linear Scale オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaAllowTransform	boolean
XmNxrtGaDebug	boolean
XmNxrtGaDirection	integer
XmNxrtGaMax	float
XmNxrtGaMin	float
XmNxrtGaOrientation	integer

Reasons
XmNxrtGaTransformCallback

F.15 Marker オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaCircularScale	widget
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageRotate	boolean
XmNxrtGaInnerRadius	float
XmNxrtGaIsShowing	boolean
XmNxrtGaMarkerType	integer
XmNxrtGaOuterRadius	float

F.16 Needle オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaCircularScale	widget
XmNxrtGaDebug	boolean
XmNxrtGaInteractionType	integer
XmNxrtGaIsShowing	boolean
XmNxrtGaLength	float
XmNxrtGaNeedleType	integer
XmNxrtGaValue	float

Reasons
XmNxrtGaValueChangedCallback

F.17 Percent ウィジェットのリソース

UIL Argument Name	Argument Type
XmNxrtGaBorder	integer
XmNxrtGaCalcMode	integer
XmNxrtGaDebug	boolean
XmNxrtGaDoubleBuffer	boolean
XmNxrtGaFrame	widget
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaRepaint	boolean
XmNxrtGaScale	widget
XmNxrtGaUseRenderList	boolean

Reasons
XmNxrtGaExposeCallback
XmNxrtGaResizeCallback

F.18 Range オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaInnerExtent	float
XmNxrtGaIsShowing	boolean
XmNxrtGaOffset	float
XmNxrtGaOuterExtent	float
XmNxrtGaScale	widget
XmNxrtGaShape	integer
XmNxrtGaStartValue	float
XmNxrtGaStopValue	float

F.19 State オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaAlignment	integer
XmNxrtGaAlignmentUseDefault	boolean
XmNxrtGaAngle	float
XmNxrtGaAngleUseDefault	boolean
XmNxrtGaCircularScale	widget
XmNxrtGaDebug	boolean
XmNxrtGaIsShowing	boolean
XmNxrtGaLineDisplay	boolean
XmNxrtGaLineInnerRadius	float
XmNxrtGaLineInnerRadiusUseDefault	boolean
XmNxrtGaLineOuterRadius	float
XmNxrtGaLineOuterRadiusUseDefault	boolean
XmNxrtGaLinePattern	integer
XmNxrtGaLineThickness	integer
XmNxrtGaOffsetUseDefault	boolean

UIL Argument Name	Argument Type
XmNxrtGaOffsetX	integer
XmNxrtGaOffsetY	integer
XmNxrtGaText	string

F.20 Switch オブジェクト

UIL Argument Name	Argument Type
XmNxrtGaCircularScale	widget
XmNxrtGaContinuous	boolean
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageRotate	boolean
XmNxrtGaIncrementValue	float
XmNxrtGaInteractionType	integer
XmNxrtGaIsShowing	boolean
XmNxrtGaMultipleTurnEnabled	boolean
XmNxrtGaMultipleTurnValue	float
XmNxrtGaRadius	float
XmNxrtGaSwitchType	integer
XmNxrtGaValue	float

Reasons
XmNxrtGaValueChangedCallback

F.21 Text オブジェクトリソース

UIL Argument Name	Argument Type
XmNxrtGaAlignment	integer
XmNxrtGaAllowWidgetPositioning	boolean
XmNxrtGaBorder	integer
XmNxrtGaBorderWidth	integer

UIL Argument Name	Argument Type
XmNxrtGaDebug	boolean
XmNxrtGaImage	pixmap
XmNxrtGaImageLayout	integer
XmNxrtGaImageMinimumSize	boolean
XmNxrtGaIsShowing	boolean
XmNxrtGaString	string
XmNxrtGaStringMode	integer
XmNxrtGaValueObject	widget
XmNxrtGaXmString	compound_string

F.22 Tick オブジェクトのリソース

UIL Argument Name	Argument Type
XmNxrtGaAutomatic	boolean
XmNxrtGaDebug	boolean
XmNxrtGaDrawLabels	boolean
XmNxrtGaIncrementValue	float
XmNxrtGaInnerExtent	float
XmNxrtGaIsShowing	boolean
XmNxrtGaLabelExtent	float
XmNxrtGaOffset	float
XmNxrtGaOuterExtent	float
XmNxrtGaPrecision	integer
XmNxrtGaPrecisionUseDefault	boolean
XmNxrtGaScale	widget
XmNxrtGaStartValue	float
XmNxrtGaStopValue	float
XmNxrtGaTickType	integer

F.23 短くなっている定数

下記の XRT/gauge の定数は、UIL の 31 文字制限に合うように短くなっています。

- XRTGA_INTERACTION_TYPE_NONE は GA_INTERACT_NONE
- XRTGA_INTERACTION_TYPE_RELATIVE_TO_MOUSE は、GA_INTERACT_RELATIVE_TO_MOUSE
- XRTGA_INTERACTION_TYPE_SNAP_TO_MOUSE は、GA_INTERACT_SNAP_TO_MOUSE

G

サンプル･コード

XRT/gauge 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。XRT/gauge の基本的なプログラムの大半は、これらの例やデモプログラムで紹介されています。

G.1 例

例プログラムは、*$XRTHOME/src/gauge/examples* にあります。

analyzer.c	Linear Gauge ウィジェットまたは LEDIcon ウィジェットを使用してスペクトラム･アナライザを表示します。
car.c	車のインストルメント･クラスタを表示します。
cdial.c	Circular Dial ウィジェットを表示します。
clock.c	時間をアナログとディジタル･モードで表示します。
color.c	Linear Gauge 上で Indicator を使用し、RGB 値を選択できます。
combo_lock.c	時間切れになる前に、Circular Dial を使用してコンビネーション･ロックを開けるゲームです。
expenses.c	家計の出費を Percent ウィジェットで表しています。
flashing.c	LEDNumerical の例です。
image.c	Linear Dial のオンとオフのイメージを表示します。
ldial.c	Linear Dial の例です。
led_icon.c	LEDIcon ウィジェットに幾つかのリソースをセットし、マウスによって Icon をオン、またはオフにすることができるようにしています。

led_num.c	LEDNumerical ウィジェットのリソースをセットしています。
led_num_text.c	LEDNumerical ウィジェットの桁数とテキストをセットしています。
lgauge.c	Linear Gauge ウィジェットのリソースをセットしています。
needle.c	Circular Gauge 内の Neelde オブジェクトのリソースをセットしています。
nim.c	LEDIcon ウィジェットを使用した NIM ゲームです。
oven.c	Circular Dial を使用した、オーブンのダイアルのシミュレーションです。
percent.c	Percent ウィジェットのリソースをセットしています。
progress.c	Circular Scale と Linear Scale を使用した二つの Progress ウィジェットです。
scoreboard.c	LEDNumerical ウィジェットを使用した野球のスコアボードです。
switch.c	Circular Dial 上の異なる Switch の型の例です。
temperature.c	温度計の形態を真似た Range オブジェクトを持つ Linear Gauge です。
ticks.c	Tick オブジェクトのリソースをセットしています。

G.2 デモ

デモ・アプリケーションは、*$XRTHOME/src/gauge/demos* にあります。各デモごとにディレクトリがあります。

flight	飛行機の計器パネルを真似たアプリケーションです。
sampler	ウィジェット別にグループ化された XRT/gauge の例のコレクションです。

索引

2

2byte フォント, 14

C

C++, 12
Center, 21, 77, 227, 249
　色の設定, 77
　サイズ, 78
　サンプル・コード, 78
　手続きとマクロ, 79
　リソースの一覧, 79
Circualr Gauge
　リソースの一覧, 28
Circular Dial, 17, 20, 137, 227, 250
　サンプル・コード, 25
　手続きとマクロ, 32
　トランスレーション, 25
　非連続モード, 18, 23
　リソースの一覧, 28
　連続モード, 18
Circular Gauge, 17, 20, 228, 247, 250
　サンプル・コード, 26
　手続きとマクロ, 32
　トランスレーション, 24
Circular Scale, 19, 229, 251
　値の増加する方向, 83
　角度, 81, 83
　サンプル・コード, 83
　手続きとマクロ, 85
　リソースの一覧, 83
Counter, 87, 230, 252
　値の設定, 89
　色, 88
　サンプル・コード, 89
　小数点, 88
　数値のロールオーバー, 88
　手続きとマクロ, 89
　トランスレーション, 89
　フォント, 88
　リセット・ボタン, 88
　リソースの一覧, 89
　枠線, 87

F

Fallback リソース, 14
Frame, 20, 22, 95, 231, 253
　位置, 96
　色, 96
　形, 96
　サンプル・コード, 96
　手続きとマクロ, 98
　リソースの一覧, 96

G

GUI ビルダ, 12, 245

I

Icon, 99, 231, 253
　色, 99
　形, 100
　サンプル・コード, 100
　手続きとマクロ, 101
　バイナリ・モード, 99
　マグニチュード・モード, 99
　リソースの一覧, 100
Indicator, 36, 103, 231, 253
　サイズ, 104
　サンプル・コード, 104
　手続きとマクロ, 107
　リソースの一覧, 105

Item, 109, 232, 254
値, 110
サンプル・コード, 110
手続きとマクロ, 110
ラベル, 110
リソースの一覧, 110

L

LEDIcon, 47, 232, 254
値の設定, 48
サイズとレイアウト, 49
最大値, 48
サンプル・コード, 50
手続きとマクロ, 55
トランスレーション, 50
バイナリ・モード, 47, 48
マグニチュード・モード, 47
リソースの一覧, 50
LEDNumerical, 57, 233, 256
値の設定, 58
クロック表示, 60
サイズ, 58
サンプル・コード, 61
数値の傾き, 60
手続きとマクロ, 65
リソースの一覧, 62
Linear Dial, 35, 36, 145, 234, 257
手続きとマクロ, 36
トランスレーション, 36
リソースの一覧, 36
Linear Gauge, 35, 36, 235, 257
サンプル・プログラム, 36
手続きとマクロ, 36
トランスレーション, 36
リソースの一覧, 36
Linear Scale, 36, 115, 236, 258
サイズと位置, 116
サンプル・コード, 117
手続きとマクロ, 119
リソースの一覧, 118
Linwe Scale
ユーザ・インタラクション, 117

M

Marker, 22, 121, 236, 258
サイズと位置, 121
サンプル・コード, 122
手続きとマクロ, 124
リソースの一覧, 123
Motif との統合, 9
MrmNcreateCallbck, 249
MtCenterText, 15

N

Needle, 22, 125, 237, 259
色, 126
形, 126
サンプル・コード, 126, 154
手続きとマクロ, 128
ユーザ・インタラクション, 126
リソースの一覧, 127

P

Percent, 67, 237, 259
calculation-by-percent, 67
calculation-by-ratio, 67
Item, 67
Scale の選択, 67
サンプル・コード, 67
手続きとマクロ, 68
トランスレーション, 67
ラベル, 67
リソースの一覧, 67

R

Range, 19, 23, 36, 129, 238, 260
Circular Scale, 129
形, 129
サイズ, 129
サンプル・コード, 129
手続きとマクロ, 129
リソースの一覧, 129

S

State, 23, 137, 238, 260

サンプル・コード, 139
線の表示, 138
手続きとマクロ, 143
配置, 137, 138
非連続モード, 137
ラベル, 137
リソースの一覧, 140
Switch, 23, 145, 239, 261
サンプル・コード, 147
手続きとマクロ, 150
非連続モード, 146
リソースの一覧, 148
連続モード, 146

T

Text, 36, 153, 240, 261
値の設定, 154
ストリング, 154
手続きとマクロ, 157
配置, 154
リソースの一覧, 155
Tick, 19, 24, 36, 159, 241, 262
形, 160
サイズ, 160
サンプル・コード, 162
精度, 161
手続きと方法, 165
配置, 160
ラベル, 161
リソースの一覧, 162

U

UIL, 249

X

XmCreateXrtGaCenter(), 79
XmCreateXrtGaCircularDial(), 32
XmCreateXrtGaCircularGauge(), 32
XmCreateXrtGaCircularScale(), 85
XmCreateXrtGaCounter(), 89
XmCreateXrtGaFrame(, 98
XmCreateXrtGaIcon(), 101
XmCreateXrtGaIndicator(), 107
XmCreateXrtGaItem(), 110
XmCreateXrtGaLEDIcon(), 55
XmCreateXrtGaLEDNumerical(), 65
XmCreateXrtGaLinearDial(), 36
XmCreateXrtGaLinearGauge(), 36
XmCreateXrtGaLinearScale(), 119
XmCreateXrtGaMarker(), 124
XmCreateXrtGaNeedle(), 128
XmCreateXrtGaPercent(), 68
XmCreateXrtGaRange(), 129
XmCreateXrtGaState(), 143
XmCreateXrtGaSwitch(), 150
XmCreateXrtGaText(), 157
XmCreateXrtGaTick(), 165
XmIsXrtGaCenter(), 80
XmIsXrtGaCircularDial(), 32
XmIsXrtGaCircularGauge(), 32
XmIsXrtGaCircularScale(), 85
XmIsXrtGaCounter(), 89
XmIsXrtGaFrame(), 98
XmIsXrtGaIcon(), 101
XmIsXrtGaIndicator(), 107
XmIsXrtGaItem(), 110
XmIsXrtGaLEDIcon(), 55
XmIsXrtGaLEDNumerical(), 65
XmIsXrtGaLinearDial(), 36
XmIsXrtGaLinearGauge(), 36
XmIsXrtGaLinearScale (), 119
XmIsXrtGaMarker (), 124
XmIsXrtGaNeedle (), 128
XmIsXrtGaPercent(), 68
XmIsXrtGaRange (), 129
XmIsXrtGaState(), 143
XmIsXrtGaSwitch(), 151
XmIsXrtGaText(), 158
XmIsXrtGaTick(), 165
XmNbackground, 28, 36, 50, 62, 67, 79, 84, 89, 97, 100, 105, 110, 118, 123, 127, 129, 148, 155, 162
XmNfontList, 89, 140, 155, 162
XmNforeground, 140, 155
XmNheight, 89, 118, 155
XmNwidth, 89, 105, 118, 123, 127, 155, 162
XmNx, 89, 118, 155
XmNxrtGaAlignment, 140, 155
XmNxrtGaAlignmentUseDefault, 140
XmNxrtGaAllowTransform, 83, 84, 117, 118
XmNxrtGaAllowWidgetPositioning, 67, 110, 155
XmNxrtGaAngle, 141
XmNxrtGaAngleUseDefault, 141
XmNxrtGaAutomatic, 160, 162

XmNxrtGaBorder, 28, 36, 51, 62, 67, 89, 156
XmNxrtGaBorderWidth, 89, 156
XmNxrtGaBottomMargin, 97
XmNxrtGaCalcMode, 67
XmNxrtGaCenterList, 30
XmNxrtGaCircularScale, 30, 77, 78, 79, 81, 123, 127, 141, 148
XmNxrtGaCircularScaleList, 30
XmNxrtGaColorMixPercentage, 58, 62
XmNxrtGaContinuous, 146, 148
XmNxrtGaCounterList, 28
XmNxrtGaDebug, 15, 28, 36, 51, 62, 67, 79, 84, 89, 97, 101, 105, 110, 118, 123, 127, 129, 141, 148, 156, 163
XmNxrtGaDecayRate, 48, 51
XmNxrtGaDecimalPlaces, 58, 62, 89
XmNxrtGaDecimalPosition, 62
XmNxrtGaDecimalSize, 62
XmNxrtGaDigitBackgroundColor, 89
XmNxrtGaDigitForegroundColor, 89
XmNxrtGaDirection, 49, 51, 83, 84, 118
XmNxrtGaDisplayMode, 48, 51
XmNxrtGaDoubleBuffer, 28, 36, 51, 62, 67
XmNxrtGaDrawFunc, 105, 123, 127, 148
XmNxrtGaDrawLabels, 160, 161, 163
XmNxrtGaExposeCallback, 28, 36, 52, 62, 67
XmNxrtGaFixedDecimal, 58, 62
XmNxrtGaFrame, 28, 36, 67
XmNxrtGaHighlightColor, 30
XmNxrtGaHighlightSelectedState, 30
XmNxrtGaIconHeight, 50, 52
XmNxrtGaIconList, 50, 52
XmNxrtGaIconSpacing, 49, 52
XmNxrtGaIconWidth, 50, 52
XmNxrtGaImage, 29, 36, 52, 63, 67, 78, 79, 89, 97, 105, 110, 123, 129, 149, 156
XmNxrtGaImageLayout, 29, 36, 52, 63, 67, 89, 97, 129, 156
XmNxrtGaImageMinimumSize, 156
XmNxrtGaImageRotate, 105, 123, 149
XmNxrtGaIncrementValue, 146, 149, 160, 163
XmNxrtGaIndicator, 36
XmNxrtGaIndicatorList, 36
XmNxrtGaIndicatorType, 106
XmNxrtGaInnerExtent, 104, 106, 129, 163
XmNxrtGaInnerRadius, 123
XmNxrtGaInteractionCallback, 29, 36, 52, 169
XmNxrtGaInteractionType, 52, 106, 126, 127, 149
XmNxrtGaIsShowing, 79, 89, 97, 106, 110, 123, 127, 129, 141, 149, 156, 163
XmNxrtGaItemList, 67
XmNxrtGaLabelExtent, 67, 110, 163
XmNxrtGaLabelFunc, 161, 163
XmNxrtGaLeadingOne, 60, 63
XmNxrtGaLeadingPlusMinus, 58, 63
XmNxrtGaLeadingZeros, 58, 63
XmNxrtGaLEDStartValue, 53
XmNxrtGaLEDValue, 48, 53
XmNxrtGaLeftMargin, 97
XmNxrtGaLength, 127
XmNxrtGaLinearScale, 36, 106
XmNxrtGaLinearScaleList, 36
XmNxrtGaLineDisplay, 141
XmNxrtGaLineInnerRadius, 31, 141
XmNxrtGaLineInnerRadiusUseDefault, 141
XmNxrtGaLineOuterRadius, 31, 141
XmNxrtGaLineOuterRadiusUseDefault, 142
XmNxrtGaLinePattern, 142
XmNxrtGaLineThickness, 142
XmNxrtGaMapCallback, 29, 36, 67, 170
XmNxrtGaMarker, 31
XmNxrtGaMarkerType, 123
XmNxrtGaMax, 84, 118
XmNxrtGaMaxIcon, 48, 53
XmNxrtGaMin, 84, 118
XmNxrtGaMultipleTurnEnabled, 149
XmNxrtGaMultipleTurnValue, 149
XmNxrtGaNeedle, 125
XmNxrtGaNeedleList, 30
XmNxrtGaNeedleType, 128
XmNxrtGaNumDigits, 58, 63, 89
XmNxrtGaOffColor, 36, 58, 63, 101
XmNxrtGaOffImage, 36, 101
XmNxrtGaOffset, 129, 160, 164
XmNxrtGaOffsetDistance, 31
XmNxrtGaOffsetType, 31
XmNxrtGaOffsetUseDefault, 142
XmNxrtGaOffsetX, 142
XmNxrtGaOffsetY, 143
XmNxrtGaOnColor, 36, 58, 63, 101
XmNxrtGaOnImage, 36, 101
XmNxrtGaOrientation, 49, 53, 119
XmNxrtGaOuterExtent, 104, 106, 129, 164
XmNxrtGaOuterRadius, 124
XmNxrtGaPrecision, 160, 161, 164
XmNxrtGaPrecisionUseDefault, 164
XmNxrtGaRadius, 78, 79, 150
XmNxrtGaRadiusSize, 84
XmNxrtGaRangeList, 29, 36
XmNxrtGaRenderList, 29, 36, 67

XmNxrtGaRepaint, 14, 29, 36, 53, 63, 67
XmNxrtGaResetAlignment, 89
XmNxrtGaResetBackgroundColor, 89
XmNxrtGaResetImage, 89
XmNxrtGaResetOffset, 88, 89
XmNxrtGaResetShape, 89
XmNxrtGaResetShow, 89
XmNxrtGaResetWidth, 89
XmNxrtGaResizeCallback, 29, 36, 53, 63, 67
XmNxrtGaRightMargin, 97
XmNxrtGaScale, 67, 129, 164
XmNxrtGaSegmentSpacing, 63
XmNxrtGaSegmentWidth, 63
XmNxrtGaShape, 36, 97, 101, 129
XmNxrtGaSizingPolicy, 49, 54
XmNxrtGaSlopeOffset, 60, 64
XmNxrtGaSpacingHorizontal, 54, 59, 64, 89
XmNxrtGaSpacingVertical, 54, 64, 89
XmNxrtGaStartAngle, 84
XmNxrtGaStartValue, 129, 160, 164
XmNxrtGaStateList, 32, 137
XmNxrtGaStopAngle, 84
XmNxrtGaStopValue, 129, 160, 165
XmNxrtGaString, 67, 157
XmNxrtGaStringMode, 154, 157
XmNxrtGaSwitch, 32
XmNxrtGaSwitchType, 150
XmNxrtGaText, 64, 143
XmNxrtGaTextList, 29, 36, 67
XmNxrtGaTickList, 30, 36
XmNxrtGaTickType, 160, 165
XmNxrtGaTopMargin, 98
XmNxrtGaTransformCallback, 83, 85, 117, 119, 169, 170
XmNxrtGaTransitionRate, 89
XmNxrtGaTransitionSteps, 89
XmNxrtGaTransitonRate, 88
XmNxrtGaUseRenderList, 30, 36, 67
XmNxrtGaUseTransition, 88, 89
XmNxrtGaValue, 64, 89, 106, 110, 128, 150
XmNxrtGaValueChangedCallback, 54, 64, 89, 106, 110, 128, 150
XmNxrtGaValueObject, 67, 157
XmNxrtGaValueType, 58, 64
XmNxrtGaXmString, 157
XmNxrtGaXOrigin, 21, 54, 85
XmNxrtGaYOrigin, 21, 54, 85
XmNy, 89, 119, 157
XmString, 14
xrt_auth, 11
XrtGaAlignment, 219
XrtGaArgValToDouble(), 215
XrtGaBorder, 219
XrtGaCalcMode, 219
XrtGaCallbackStruct, 224
XrtGaDirection, 220
XrtGaDisplayMode, 220
XrtGaDoubleToArgVal(), 215
XrtGaDrawData, 181, 224
XrtGaDrawPolyTransform(), 182, 216
XrtGaEnableImageType(), 216
XrtGaFillPattern, 220
XrtGaFrame, 95
XrtGaGetUserDrawInfo(), 181
XrtGaImageLayout, 220
XrtGaIndicator, 103, 184
XrtGaIndicatorType, 220
XrtGaInteractionCallbackStruct, 224
XrtGaInteractionType, 221
XrtGaItem, 109
XrtGaLinearScale, 115
XrtGaLinePattern, 221
XrtGaMap(), 170, 216
XrtGaMapCallbackStruct, 224
XrtGaMapResult, 225
XrtGaMarker, 121, 185
XrtGaMarkerType, 221
XrtGaNeedle, 185
XrtGaNeedleType, 221
XrtGaOffsetType, 221
XrtGaPolyDrawStruct, 182, 225
XrtGaRange, 129
XrtGaReason, 222
XrtGaSetXmStringConverter(), 216
XrtGaShape, 222
XrtGaSizingPolicy, 222
XrtGaState, 137
XrtGaStringMode, 223
XrtGaSwitch, 145, 185
XrtGaSwitchType, 223
XrtGaText, 153
XrtGaTick, 159
XrtGaTickType, 223
XrtGaTransformCallbackStruct, 225
XrtGaUnmap(), 170, 216
XrtGaUserDrawInfo, 226
XrtGaValueChangedCallbackStruct, 226
XrtGaValueType, 223
XRTHOME, 9
XrtPDSCmpDoubles(), 192, 194

XrtPDSCmpFloats(), 192, 194
XrtPDSCmpInts(), 194
XrtPDSCmpString(), 192
XrtPDSCmpStrings(), 194
XrtPDSCmpStringsIgnoreCase(), 194
XrtPDSImage, 189
    イメージ・ファンクション, 190
    イメージ型の登録, 190
    作成, 189
    任意のイメージ型, 190
XrtPDSImageCopy(), 194
XrtPDSImageCreate(), 194
XrtPDSImageCreateProc, 190
XrtPDSImageDestroy(), 194
XrtPDSImageFuncs, 190
XrtPDSImageGetClipMask(), 194
XrtPDSImageGetColorDepth(), 194
XrtPDSImageGetFilename(), 194
XrtPDSImageGetHeight(), 194
XrtPDSImageGetImage(), 194
XrtPDSImageGetWidth(), 194
XrtPDSImageHeaderCompareProc, 190
XrtPDSImageIsClipMaskSet(), 194
XrtPDSImageIsImage(), 194
XrtPDSImageIsSet(), 194
XrtPDSImageNameCompareProc, 190
XrtPDSImageOutputProc, 190
XrtPDSList, 190
    アイテムの検索と取得, 190
    アイテムの除去と削除, 190
    アイテムの追加, 190
    エレメントの移動, 193
    作成, 190
    サンプル・コード, 190
    ユーザ・データ, 194
    リストのソート, 193
    リスト比較ルーチン, 192
XrtPDSListAdd(), 190
XRTPDSListAdd(), 194
XrtPDSListAppend(), 190, 194
XrtPDSListCreate(), 190, 194
XrtPDSListDelete(), 194
XrtPDSListDeleteALL(), 194
XrtPDSListDestroy(), 190, 194
XrtPDSListDestroyAll(), 190
XrtPDSListDestroyObject(), 194
XrtPDSListDestroyXtPoiter(), 194
XrtPDSListDuplicate(), 190, 194
XrtPDSListFind(), 191, 194
XrtPDSListGetCompareProc(), 194
XrtPDSListGetItem(), 191, 194
XrtPDSListGetItemCount(), 191, 194
XrtPDSListGetItems(), 191, 194
XrtPDSListGetItemSize(), 194
XrtPDSListGetUserData(), 194
XrtPDSListInsertAfter(), 194
XrtPDSListInsertBefore(), 194
XrtPDSListInsetBefore(), 190
XrtPDSListIsList(), 190, 194
XrtPDSListIsSorted(), 194
XrtPDSListLookupItem(), 191, 194
XrtPDSListMoveItemsAfter(), 194
XrtPDSListMoveItemsBefore(), 193, 194
XrtPDSListRemove(), 190, 194
XrtPDSListRemoveAll(), 190, 194
XrtPDSListSearchBackward(), 191
XrtPDSListSearchForward(), 191, 194
XrtPDSListSearchReverse(), 194
XrtPDSListSetCompareProc(), 191, 194
XrtPDSListSetDestroyProc(), 194
XrtPDSListSetItem(), 194
XrtPDSListSetUserData(), 194
XrtPDSListSort(), 194
XtConvertAndStore(), 186
XtNxrtGaExposeCallback, 172
XtNxrtGaMapCallback, 170
XtNxrtGaResizeCallback, 172

あ

アクション・ルーチン
    ActionEnd(), 169
    ActionMove(), 168
    ActionStart(), 167
    ResetCounter(), 170
    TranslateEnd(), 170
    TranslateStart(0, 169
    TranslateUpdate(), 170
    ZoomEnd(), 170
    ZoomStart(), 169
    ZoomUpdate(), 170
アプリケーションの許可, 11

か

環境変数, 9

き

共用ライブラリ, 190

こ

コールバック
    Expose, 172
    Resize, 172
    値変更, 171
コンパイル, 11

た

ダイナミック・ライブラリ, 190

て

データ型, 219
デバッグ, 15
デモプログラム, 265

ひ

描画ファンクション, 181

ま

マクロ, 14, 215

ゆ

ユーザ・インタラクション, 167
    値変更のコールバック, 171
    イベント処理, 171
    開始, 167
    カスタマイズ, 167
    デフォルト, 167
    不可, 168
    マッピング, 170
    リサイズ, 172

り

リソース
    継承されたリソース, 243
    リソース・ファイル, 245
リソースの型とクラス, 227
リソースの変更, 14
リソース・ファイル, 12
    構文, 245