Solaris X Window System 開発ガイド

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650–960–1300

Part Number 806-2735-10
2000 年 3 月

原典: *Solaris X Window System Developer's Guide*

Part No: 806-1363-10

Revision A

Adobe PostScript

目次

はじめに

このマニュアルでは、Solaris™ X サーバーについて詳しく説明します。サーバーアーキテクチャの概要や詳しい情報の記載場所についても説明しています。

また、Solaris X サーバーとのインタフェースに関与するソフトウェア開発者向けの詳細な情報も記載してあります。

対象読者

Solaris X 環境でプログラムを作成するにはツールキットの使用が不可欠であり、場合によってはサーバーやそのプロトコルとのインタフェースをとることも必要です。このプロトコルとツールキットについては他のマニュアルで解説されています(11ページの「関連マニュアル」を参照)。次の項目について詳細な情報が必要な方は、このマニュアルをお読みください。

- Solaris X サーバーの機能
- X コンソーシアムサンプルサーバーとの相違点と拡張機能
- DPS イメージ処理システム
- サポートされている表示デバイス
- サーバー接続用の認証方法とプロトコル

前提条件

このマニュアルでは、読者はプログラミングの経験と知識があるか、あるいは以下の内容に関する適切なマニュアルを読まれていることを前提としています。

- Solaris 7 およびその互換バージョン
- X Window System™
- C プログラミング言語
- PostScript™
- Display PostScript™ システム (DPS)
- `olwm` ウィンドウマネージャ
- XView™ ツールキット

内容の紹介

このマニュアルは先頭から順に読んでも、必要な章だけ拾い読みしてもかまいません。概要を知るための、また参照のためのドキュメントのどちらにもご利用いただけます。

第 1 章では、Solaris X サーバーのアーキテクチャ、X と DPS の拡張機能、Sun が X コンソーシアムライブラリと拡張機能に対して行った強化、カラーに関する問題点、およびサーバー上で実行できるアプリケーションについて説明します。

第 2 章では、Solaris 特有の DPS 機能と、標準 DPS の拡張機能として提供されるオペレータの構成について説明します。

第 3 章では、Solaris 環境におけるビジュアルを説明し、ビジュアルによるウィンドウプログラミングのヒントを示します。

第 4 章では、提供されるフォントセットと、フォントを管理する方法について説明します。

第 5 章では、透明オーバーレイウィンドウと比較しながら、サーバーオーバーレイについて説明します。

第 6 章では、透明オーバーレイウィンドウのための、アプリケーションプログラムインタフェース (API) を説明します。

第 7 章では、Solaris 環境のセキュリティ機能を説明します。

付録 Aでは、Solaris 環境での基準デバイスとして提供されるグラフィックスデバイスを説明します。

関連マニュアル

Solaris 環境でアプリケーションを書くための詳しい情報については、次のマニュアルを参照してください。

- 『*ToolTalk* ユーザーズガイド』
- 『*OpenWindows* デスクトップリファレンスマニュアル』
- 『*Solaris X Window System* リファレンスマニュアル』
- 『*X Server Device Developer's Guide*』
- 『フォントの管理』
- 『*XView* 開発ガイド』
- 『*OLIT* プログラミングの手引き』
- 『*OLIT* リファレンスマニュアル』

X に関する次のマニュアルは、お近くの書店でお求めになれます。

- 『*XView ver.3* リファレンス・マニュアル』(ソフトバンク発行)
- 『*XView* プログラミング・マニュアル』(ソフトバンク発行)
- 『*Xlib* リファレンス・マニュアル』(ソフトバンク発行)
- 『*Xlib Programming Manual*』(O'Reilly & Associates 発行)
- 『*X* プロトコル・リファレンス・マニュアル』(ソフトバンク発行)
- 『*X11 Release 5* 増補版』(ソフトバンク発行)
- 『*X* ツールキット・イントリンシクス・プログラミング・マニュアル』(ソフトバンク発行)
- 『*X Window System, Third Edition*』(Digital Press 発行)
- 『*The X Window System Server, X Version 11, Release 5*』(Digital Press 発行)

PostScript と DPS に関する次のマニュアルは、お近くの書店でお求めになれます。

- 『*PostScript Language Reference Manual, Second Edition*』(Adobe® 社発行)

- 『*PostScript Language Tutorial and Cookbook*』(Adobe 社発行)
- 『*Programming the Display PostScript System with X*』(Adobe 社発行)
- 『*PostScript プログラム・デザイン*』(アスキー発行)
- 『*Adobe Type I Font Format*』(Adobe 社発行)

Sun のマニュアルの注文方法

専門書を扱うインターネットの書店 Fatbrain.com から、米国 Sun Microsystems™, Inc. (以降、Sun™ とします) のマニュアルをご注文いただけます。

マニュアルのリストと注文方法については、`http://www1.fatbrain.com/documentation/sun` の Sun Documentation Center をご覧ください。

Sun のオンラインマニュアル

`http://docs.sun.com` では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索をおこなうこともできます。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–1 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

ただし AnswerBook2™ では、ユーザーが入力する文字と画面上のコンピュータ出力は区別して表示されません。

コード例は次のように表示されます。

- C シェルプロンプト

```
system% command y|n [filename]
```

- Bourne シェルおよび Korn シェルのプロンプト

```
system$ command y|n [filename]
```

- スーパーユーザーのプロンプト

```
system# command y|n [filename]
```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。
- このマニュアルでは、「IA」という用語は、Intel 32 ビットのプロセッサアーキテクチャを意味します。これには、Pentium、Pentium Pro、Pentium II、Pentium II Xeon、Celeron、Pentium III、Pentium III Xeon の各プロセッサ、および AMD、Cyrix が提供する互換マイクロプロセッサチップが含まれます。

第 1 章

Solaris X サーバーの概要

この章では、Solaris X サーバーの概要を説明します。Solaris X サーバーにより、Solaris 製品で X Window System のクライアント/サーバーモデルが実現されます。この章では、次の内容について説明します。

- 今回のリリースで Solaris X サーバーに追加された新機能
- X コンソーシアムがサポートする拡張機能や Display PostScript 拡張機能など、Solaris X サーバーの機能
- サポートされる X11 アプリケーションとサポートされない X11 アプリケーション
- OpenWindows™ のディレクトリ構造

Solaris X サーバーの概要

Solaris X サーバー、`XSun` は、Display PostScript (DPS) イメージシステム拡張機能を持つ X コンソーシアムの X11R6 サンプルサーバー、X コンソーシアムのその他の X 拡張機能、および Sun が独自に追加した機能からなっています。Solaris X サーバーは、共通デスクトップ環境 (CDE) の基礎となるもので、CDE デスクトップの下に位置付けられます。このサーバーは、クライアントアプリケーション、表示ハードウェア、入力デバイス間の通信を処理します。デフォルトでは、Solaris X サーバーでは CDE dtlogin およびウィンドウマネージャ (`dtwm`) が実行されますが、ICCCM (クライアント間通信規約マニュアル) 準拠の X Window System マネージャであれば、どんなものでもこのサーバーで実行できます。ソフトウェア開発者は、Xlib ライブラリだけでなく、Motif ツールキットや Xt ツールキットなど、さまざまなツールキットを使用して、Solaris 環境用のアプリケーションを作成できます。

図 1–1 は、Solaris X サーバー、複数のデスクトップクライアントアプリケーション、ディスプレイ、入力デバイス間の関係を示します。

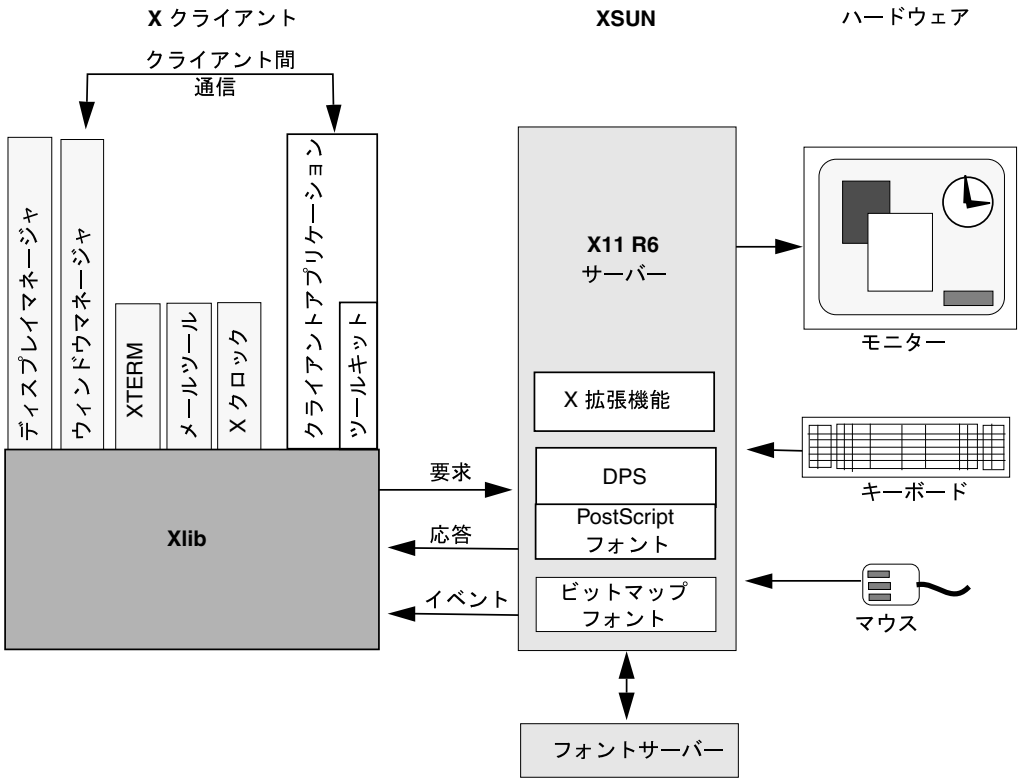


図 1–1 Solaris X サーバー

X11R6 サンプルサーバー

X コンソーシアム の X11R6 サンプルサーバーは、Solaris X サーバーの重要な構成要素です。X11R6 サンプルサーバーは、移植性を持つように設計され実装されたので、クライアントアプリケーションではハードウェアの違いを意識する必要がありません。このサンプルサーバーはすべての描画を処理し、デバイスドライバとのインタフェースにより入力を受け取り、見えない画面領域のメモリ、フォント、カーソル、カラーマップを管理します。

サンプルサーバーには、次の各部、つまり層が含まれています。

- デバイス独立層 (DIX) – クライアント要求のディスパッチ、イベント待ち行列の管理、クライアントへのイベント分配、表示用データ構造の管理を行います。この層には、グラフィックスハードウェア、入力デバイス、またはホストオペレーティングシステムに依存しない機能が含まれます。
- デバイス依存層 (DDX) – ピックスマップ、クリッピング処理用領域、カラーマップ、画面、フォント、グラフィックスコンテキストを生成し、操作します。また、DDX 層は入力デバイスからイベントを収集し、DIX 層に渡します。この層には、サーバーで使用するグラフィックスハードウェアと入力デバイスに依存するルーチンが入っています。
- オペレーティングシステム層 (OS) – クライアント接続と接続認証機構を管理し、メモリの割り当て、解放用ルーチンを提供します。OS 層には、ホストのオペレーティングシステムに依存する機能が入っています。
- フォント管理用ライブラリ – フォント管理用ライブラリにより、サーバーはフォーマットの異なるフォントファイルを使用したり、X フォントサーバーからフォントをロードしたりできます。サーバーのフォント機能については、第 4 章で説明します。

図 1–2 は、サーバーの構造を示します。このマニュアルでは、サーバーは Solaris X サーバーを意味し、サンプルサーバーは X コンソーシアムの X11R6 サンプルサーバーを意味するので注意してください。

サーバーのアーキテクチャ

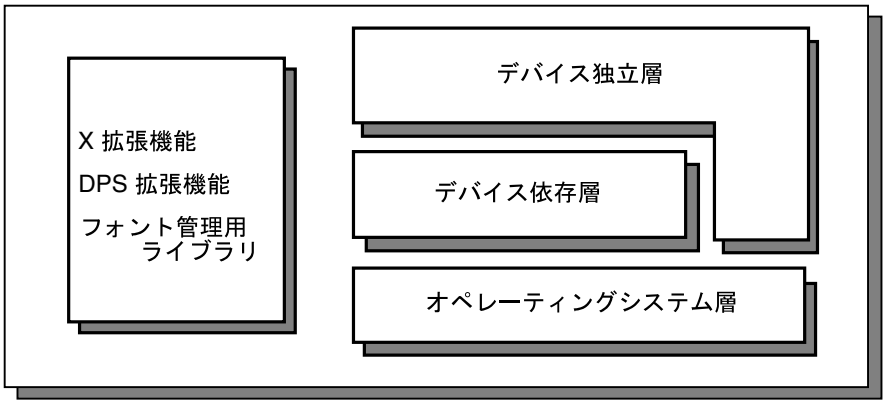


図 1–2　Solaris X サーバーのアーキテクチャ

DPS 拡張機能

Solaris X サーバーには、X11R6 サンプルサーバーの他に、Display PostScript システムが組み込まれています。DPS は、PostScript イメージングモデルと Adobe Type Library へのアクセスを X アプリケーションに提供します。Display PostScript システムは、クライアント/サーバーネットワークアーキテクチャの一部として、X Window System の拡張機能により組み込まれています。この拡張機能は、*DPS/X*[1] と呼ばれることがあります。

DPS システムには、X サーバーの拡張機能として PostScript インタプリタが実装されており、各アプリケーションがクライアントとなります。アプリケーションは、1 つのオペレータを呼び出すことにより PostScript 言語コードをサーバーに送信するので、サーバーからはデータを出力引数形式で返すことができます。DPS のクライアント/サーバー通信は、X Window System が提供する低水準通信プロトコルを使用して透過的に実行されます。DPS システムについての詳細は、第 2 章を参照してください。

X コンソーシアム拡張機能

Solaris X サーバーでは、X コンソーシアムによって定義された X 拡張機能がサポートされます。これらの拡張機能については、次の各節で簡単に説明します。各節には、拡張機能ごとの仕様名の他に、(`ftp.x.org` マシン上の) 関連ファイル名を括弧で囲んで掲載してあります。標準 X 拡張機能のメカニズムについての詳細は、『*The X Window System Server*』と『*Xlib Programming Manual*』を参照してください。

以降の各節で説明する X コンソーシアムの X11 規格は、World Wide Web 上のシステムで簡単に参照できます。URL は `http://www.rdg.opengroup.org` です。X11 マニュアルは、`ftp.x.org` マシン上の `/pub/R6untarred/mit/doc/extensions` ディレクトリに入っています。ファイル転送プロトコル (`ftp`) を使用して、このシステムからファイルをダウンロードしてください。`ftp` の使用方法に関するヘルプが必要な場合は、`ftp(1)` のマニュアルページを参照してください。自分のシステムがインターネットに接続されているかどうかは、システム管理者に問い合わせてください。

1. この節は、Adobe Systems Incorporated の『*Programming the Display PostScript System with X*』(Addison-Wesley Publishing Company, Inc.、1993 年刊) の第 2 章をもとに記述してあり、著作権所有者の承諾を得て使用しています。

XInput 拡張機能

XInput 拡張機能は、MIT の X コンソーシアム標準「X11 入力拡張機能プロトコル仕様」(/pub/X11/R6.1/xc/doc/specs/Xi/protocol.ms) を Sun が実現したものです。この拡張機能によって、代替入力デバイス (キーボードおよび ポインタ以外のデバイス) に対するアクセスを制御することが可能になり、クライアントプログラムは相互に独立して、またコアデバイスに依存することなく、各デバイスからの入力を選択することができます。

ダブルバッファ拡張機能

ダブルバッファ拡張機能 (DBE) は、X コンソーシアムの規格を Sun が実現したものです。ダブルバッファリングは、アプリケーションが完全にレンダリングされたフレームだけを表示できるようにすることで、ちらつきがないアニメーション機能を提供します。フレームは非表示バッファにレンダリングされた後、表示バッファに移動されます。

SHAPE 拡張機能

SHAPE 拡張機能は、X コンソーシアム標準「X11 非矩形ウィンドウ形状拡張機能」(shape.ms) を Sun が完全に実現したものです。この拡張機能により、X プロトコルの範囲内で任意のウィンドウおよび境界線の形状を作成することができます。

共有メモリ拡張機能

MIT-SHM (共有メモリ) 拡張機能は、X コンソーシアムの実験的な「MIT 共有メモリ拡張機能」(mit-shm.ms) を Sun が完全に実現したものです。この拡張機能により、クライアントは、実際の画像データを共有メモリに格納するという方法でメモリの XImages およびピックスマップを共有できます。メモリの共有によって Xlib プロセス間通信チャネルを介したデータ転送が不要になるため、大きな画像の場合はシステムパフォーマンスが向上します。ただし、この拡張機能が有用とされるのは、クライアントアプリケーションがサーバーと同じマシン上で実行される場合だけです。

XTEST 拡張機能

XTEST 拡張機能は、MIT の X コンソーシアムの標準案「*X11* 入力統合拡張機能案」(xtest1.mm) を Sun が完全に実現したものです。この拡張機能により、ユー

ザーがいなくても、クライアントでユーザー入力を生成したり、ユーザー入力動作を制御したりできます。この拡張機能を使用するには、サーバーの DDX 層を変更する必要があります。

その他の拡張

MIT-SUNDRY-NONSTANDARD 拡張機能は MIT で開発されたため、`ftp.x.org` のマシン上での標準ではありません。この拡張機能は、X11R3 以前のクライアントが出すさまざまな間違ったプロトコル要求を処理します。特定の誤った要求が処理されるようにバグ互換モードを有効にするか、誤った要求についてはエラーを返すようにバグ互換モードを無効にする要求を出します。また、この拡張機能は、現在のモードの状態を取得する要求も出します。

この拡張機能は、xset で動的に有効、無効を切り替えることも、`openwin` でサーバー起動時に指定することもできます。詳細は、xset(`1`) と openwin(`1`) のマニュアルページ (特に `-bc` オプション) を参照してください。

XC-MISC

この X コンソーシアムの標準拡張機能により、アプリケーションは XID を再利用できます。アプリケーションによっては、XID の作成と破棄がただちに行われるため、決められた XID の範囲を超える場合があります。ほとんどのアプリケーションではこの拡張機能は必要ありません。この仕様は、`/pub/X11/xc/doc/specs/Xect/xc-misc.ms` にあります。

X イメージング拡張機能

X イメージング拡張機能 (X Imaging Extension、XIE) は、X コンソーシアム標準を Sun が実装したものです。

AccessX

また、Solaris X サーバーでは、American Disabilities Act (ADA) 準拠のキーボード機能もサポートされます。これらの機能は、AccessX というサーバー拡張機能を通じて利用できます。AccessX 拡張機能は、スティッキーキー、スローキー、トグルキー、マウスキー、バウンスキー、リポートキーなどの機能を提供します。クライアントプログラム `accessx` を使用して、これらの機能の有効と無効を切り替えてください。`accessx` クライアントは、トグルキー、バウンスキー、リピート

キーとその設定を制御します。スティッキーキー、スローキー、マウスキーは、Shift などのキーを使用して有効にすることができます。AccessX の使用方法についての詳細は、『*Solaris* ユーザーズガイド』を参照してください。

`accessx` を実行する前に、`/usr/openwin/lib/app-defaults/accessx.uid` に UIDPATH 環境変数を設定してください。

`accessx` クライアントは、`SUNWxwacx` パッケージの一部です。このクライアントをインストールするには、`All Cluster` をインストールする必要があります。

共有メモリのトランスポート

Solaris X サーバーには、Sun の拡張機能 SUN_SME が組み込まれています。これは、Sun が共有メモリトランスポートメカニズムを実装したものです。この拡張機能により、共有メモリ経由でクライアント要求をサーバーに送信できます。共有メモリは、クライアント要求専用です。サーバーからの応答とイベントは、デフォルトのトランスポートメカニズム経由で送信されます。このトランスポートメカニズムを有効にするには、`DISPLAY` 環境変数を `:x.y` に設定します。この場合、x はディスプレイ番号で、y は画面番号です。また、環境変数 `XSUNTRANSPORT` を `shmem` に設定してください。環境変数 `XSUNSMESIZE` を希望の K バイト数に設定すれば、セグメントのサイズを設定できます。デフォルトでは、`XSUNSMESIZE` は 64 に設定されます。

ビジュアルオーバーレイウィンドウ

Solaris X サーバーには、透明オーバーレイウィンドウを有効にするアプリケーションプログラマ用インタフェース (API) も組み込まれています。オーバーレイとは、グラフィックスを描画できるピクセルバッファ (物理的に、またはソフトウェアによりシミュレート) です。アプリケーションでは、オーバーレイを使用してディスプレイウィンドウに一時イメージを表示できます。オーバーレイ API についての詳細は、第 5 章および第 6 章を参照してください。

X11 ライブラリ

表 1–1 に X ライブラリを示します。これらライブラリの構成要素である `.so` と `.a` ファイルは、`/usr/openwin/lib` にあります。

表 1–1　X11 ライブラリ

ライブラリ	説明	X コンソーシアムからの入手	Sun での改良点
libX11	Xlib	可	マルチスレッド対応 ロケールの動的ロード /usr/openwin を含むサーチパス 新しいキーシム
libXau	X 認証ライブラリ	可	なし
libXaw	アテナ widget セットライブラリ	可	なし
libXext	X 拡張ライブラリ	可	バグの修正、透明オーバーレイ
libXinput	以前の入力拡張機能とのバイナリ互換ライブラリ	不可	Sun ライブラリ
libXi	Xinput 拡張ライブラリ	可	バグの修正 OpenWindows X 拡張機能のサポート
libXmu	X 各種ユーティリティライブラリ	可	/usr/openwin を含むサーチパス
libXol	OLIT ライブラリ	不可	Sun の製品 — OLIT に関するマニュアルは「はじめに」を参照 (USL からは可)
libXt	Xt Intrinsics ライブラリ	可	なし
libxview	XView ライブラリ	可	X コンソーシアムに寄贈した Sun の製品 X11R6 の libxview に入っていないバグの修正を実施

64 ビットの X11 ライブラリ

/usr/openwin/lib/sparcv9 には 64 ビット Solaris のインストール用に、以下のような 64 ビットバージョンの共有ライブラリがあります。

libX11.so.4

libXext.so.0

libICE.so.6

libSM.so.6

libXt.so.4

libXaw.so.5

libXmu.so.4

libXtst.so.1

libXi.so.5

libXinput.so.0

libdps.so.5

libdga.so.1

libowconfig.so.0

さらに、/usr/openwin/lib/sparcv9 には以下のようなプログラマ用の 64 ビットバージョンの lint ライブラリがあります。

llib-lX11.ln

llib-lXaw.ln

llib-lXext.ln

llib-lXmu.ln

llib-lXt.ln

Solaris X サーバーで実行されるアプリケーション

Solaris X サーバーでは、次のアプリケーションを実行できます。

- 次のツールキットを使用して作成されたアプリケーション

- OpenWindows ツールキット: OLIT と XView
- Motif ツールキット
- Xt ツールキット

- X プロトコルに準拠して作成したアプリケーション
- SPARC SunOS 4.0/4.1 およびその互換バージョン上でコンパイルした OpenWindows バージョン 3 の X11 アプリケーション

注 - OpenWindows バージョン 3 の X11 アプリケーションは、システムのバイナリ互換パッケージに従う必要があります。詳細は『バイナリ互換性ガイド』を参照してください。

- 次のインタフェースを使用して作成したアプリケーションはサポートされません。
  - TNT、NeWS、XVPS
  - SunView、SunWindows、Pixrect

サポートされている X11 アプリケーション

Solaris X 環境には、X コンソーシアムから入手できる次のようなクライアントアプリケーションがあります。

- `xterm` 端末エミュレータ
- `twm` ウィンドウマネージャ
- `xdm` ディスプレイマネージャ
- `bitmap` ビットマップエディタ
- `xfd` フォントディスプレイユーティリティ
- `xauth` アクセス制御プログラム
- `xhost` アクセス制御ユーティリティ
- `xrdb` リソース制御プログラム
- `xset` ユーザープリファレンス設定プログラム
- `xsetroot` ルートウィンドウの外観設定ユーティリティ
- `xmodmap` キーボード制御ユーティリティ

- xlsfonts サーバーフォント一覧表示ユーティリティ
- xfontsel フォント選択ユーティリティ
- xlswins ウィンドウ一覧表示ユーティリティ
- xwininfo ウィンドウ情報ユーティリティ
- xlsclients クライアントアプリケーション情報ユーティリティ
- xdpyinfo サーバー情報表示ユーティリティ
- xprop ウィンドウとフォントの属性ユーティリティ

サポートされていないアプリケーション

次のアプリケーションおよびライブラリは、すべて X コンソーシアムから提供されているものであり、サーバー上で動作しますが、 Sun 社からは提供もサポートもされていません。

- Andrew、InterViews
- uwm および wm ウィンドウマネージャ
- CLX Common Lisp インタフェース
- *contrib* X コンソーシアムクライアント

OpenWindows のディレクトリ構造

図 1–3 は、OpenWindows のディレクトリ構造を示します。これには、Solaris X サーバーの実行可能ファイルと X11 コアディストリビューションライブラリが含まれています。/openwin/etc は /openwin/share/etc へのシンボリックリンクで、/openwin/include は /openwin/share/include へのリンクで、/openwin/man は /openwin/share/man へのリンクなので注意してください。/share ディレクトリには、アーキテクチャに依存しないファイルが入っています。

/openwin/lib 内の X11 ライブラリについての詳細は、21ページの「X11 ライブラリ」を参照してください。

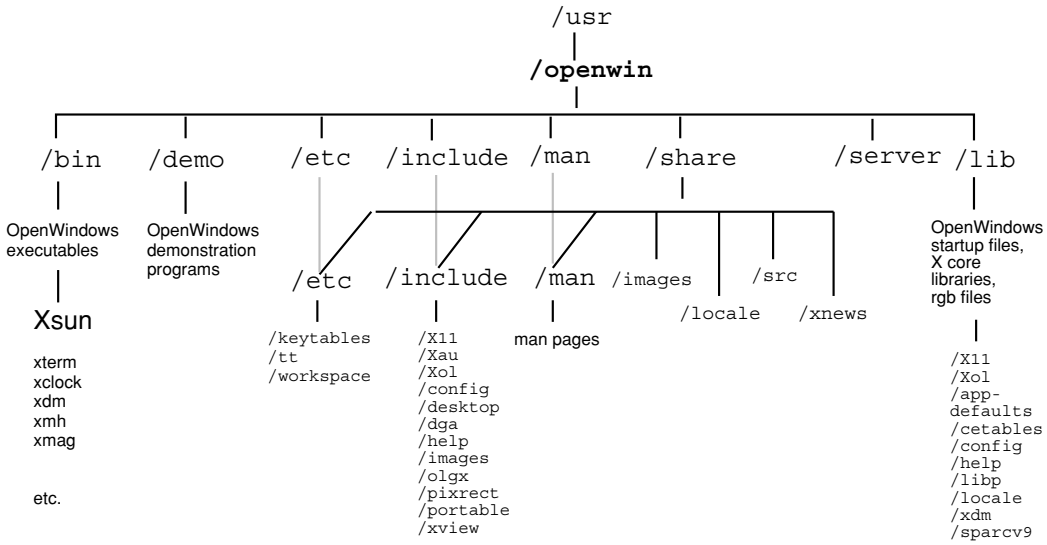


図 1–3　OpenWindows のディレクトリ構造

表 1–2 では、OpenWindows ディレクトリ構造の最上位ディレクトリの内容を簡単に説明します。

表 1–2　OpenWindows のディレクトリ

ディレクトリ	サブディレクトリ	内容
/etc	/keytables	米国と各国対応のキーテーブル、keytable.map
	/tt	ToolTalk® のデータファイル
	/workspace	/patterns (.xbm ファイルと属性)
/include	/X11	X11 ヘッダファイル、/DPS、/Xaw、/Xmu、/bitmaps、/extensions
	/Xau	/include/X11 へのシンボリックリンク
	/Xol	OLIT ヘッダファイル
	/config	generic.h ヘッダファイル
	/desktop	分類機構ヘッダファイル

表 1–2 OpenWindows のディレクトリ 続く

ディレクトリ	サブディレクトリ	内容
	/dga	dga.h ヘッダファイル
	/help	libhelp ヘッダファイル
	/images	各種ビットマップファイル
	/olgx	olgx ヘッダファイル
	/pixrect	Pixrect ヘッダファイル
	/portable	ヘッダファイル c_varieties.h と portable.h
	/xview	XView ヘッダファイル
/lib	/X11	サーバーサポートファイル、/fonts、DPS .upr ファイル
	/Xol	OLIT データファイル
	/app-defaults	X アプリケーションのデフォルトファイル
	/cetables	分類機構テーブル
	/config	imake ファイル
	/help	/locale/C/help へのシンボリックリンク
	/libp	プロファイルライブラリ
	/locale	ロケールライブラリ (/C、/iso_8859_1)
	/xdm	Xdm 構成ファイル
	/sparcv9	64 ビットの X ライブラリ

表 1–2　OpenWindows のディレクトリ　続く

ディレクトリ	サブディレクトリ	内容
/man	/man1, /man1m	OpenWindows コマンドのマニュアルページ
	/man3	XView、OLIT、Xt、Xlib などのライブラリのマニュアルページ
	/man4	AnswerBook のマニュアルページ
	/man5	ファイル形式のマニュアルページ
	/man6	デモのマニュアルページ
	/man7	コマンド以外のマニュアルページ
/server	内部使用のためのサーバー専用ファイル	
/share	/etc	/etc 内でのファイルの位置
	/images	/PostScript、/fish、/raster
	/include	/include 内のファイルの位置
	/locale	/lib/locale 内のファイルの位置
	/man	/man 内のファイルの位置
	/src	/dig_samples、/extensions、/fonts、/olit、 /tooltalk、/xview
	/xnews	/client

X11 プログラミングに関する注意

X11 プログラミングに関する一般的な問題について以下に説明します。

Compose キーのサポート

Xlib の OpenWindows のバージョンは、`XLookupString` の呼び出しによる Compose キー処理をサポートしています。

注 [IA] - IA キーボード上では、Ctrl-Shift-F1 キーを使用して Compose キーの機能を実行できます。

NumLock キーのサポート

Xlib の OpenWindows バージョンは、`XLookupString` の呼び出しによる NumLock キー処理をサポートしています。この変更は、XView、OLIT、Motif、X アプリケーション内の NumLock 処理には影響しません。

注 [IA] - IA キーボードの場合、NumLock キーはキーボードのキーパット部分の最上部にあります。

カラー名データベース

ASCII カラー名と RGB カラー値との対応関係がカラー名データベースによって与えられます。このデータベースは、カラー表示プログラムの移植性を向上させ、プログラミング作業を容易にする効果があります。ただし、この対応関係は主観的に決定したもので、科学的な客観性に基づいたものではないということに注意してください。

このデータベースのソースは `/usr/openwin/lib/X11/rgb.txt` です。このファイルは、X コンソーシアムの X11R6 で提供されるファイルと同じものです。rgb.txt をコンパイルすることによって、dbm(3) データベースファイル rgb.dir および rgb.pag が作成されます。サーバーは、最初にスタートアップされた時点

で、それらのファイルの内部表現を生成します。この内部表現を参照することによって、カラー名からカラー値へのマッピングが実行されます。

X11 クライアントは、カラー名からカラー値へのマッピングに XLookupColor または XAllocNamedColor を使用します。これらのルーチンに渡されるカラー名文字列は、それがデータベース中で検索される前に、小文字に変換されます。

カラーに関する推奨事項

Solaris X サーバーのカラーサポート機能を使用するプログラマは以下の点について考慮する必要があります。プログラムの移植性および色の共有を最大限可能にするためのヒントを以下に示します。

- デフォルト PseudoColor カラーマップ中の白と黒が特定の位置にあると仮定することはできません。必ず XAllocColor を用いて描画のためのピクセルを割り当てます。

注 - 白と黒が特定のピクセル位置にあると仮定しないことが、プログラミングの上で重要です。Solaris X サーバーの今後のバージョンや他ベンダのサーバーでは、これらの色の位置が現在のサーバーと異なることも考えられます。移植性と互換性をできるだけ高くするために、X11 クライアントは常に XAllocColor 関数で描画用の色を割り当てるようにしてください。

- ビジュアルを使用する前に、サポートされているビジュアルの種類をすべて確認しなければなりません。これには XGetVisualInfo または XMatchVisualInfo を使用できますが、XGetVisualInfo はクラスおよびデプスが同じビジュアルを区別できるため、その使用を推奨します。
- カラーマップフラッシングを減少させるため、最初にデフォルトカラーマップから色を割り当てる方法が一般に有効とされます。専用カラーマップを作成するのは、この割り当てに失敗した場合だけに限ってください。
- ポータブルな X11 カラー表示クライアントの作成方法については 51ページの「ビジュアルに関するプログラミング上のヒント」の節を参照してください。

関連マニュアル

X と X Window System のさまざまな側面に関して多数の書籍が発行されています。X Window System についての詳細は、11ページの「関連マニュアル」の推奨マ

ニュアルを参照してください。これらのマニュアルは、SunExpress または最寄りの書店で入手できます。Solaris X サーバーと X コンソーシアムサンプルサーバーについての詳細は、次のマニュアルページを参照してください。

- `Xsun(1)` – Solaris X サーバー
- `Xserver(1)` – X コンソーシアムサンプルサーバー
- `openwin(1)` – OpenWindows 起動コマンド

第 2 章

DPS の機能と拡張

この章では、Solaris X サーバーの Display PostScript (DPS) 拡張機能について、次の項目を簡単に説明します。

- DPS システムの概要
- DPS の Solaris フォント拡張
- DPS のセキュリティ
- DPS 合成オペレータ

DPS について

Display PostScript システムは、プリンタやタイプセッタ[1] で標準となっている PostScript 言語と同じイメージ処理モデルを使用して、コンピュータの画面上にグラフィック情報を表示します。X アプリケーションは PostScript 言語を使用することにより、高い精度で線分や曲線を描画したり、イメージの回転や拡大・縮小処理を行ったり、入力データをグラフィックオブジェクトとして扱ったりできます。また、Display PostScript システムを使用する X アプリケーションは、Adobe タイプライブラリをすべて使用できます。

PostScript を搭載したプリンタやタイプセッタの特長は、デバイスや解像度に依存しない点です。対話処理用の画面でも、Display PostScript システムを使用すればこの恩恵にあずかります。DPS システムを利用するアプリケーションは、DPS システ

1. この節は、Adobe 社『*Programming the Display PostScript System with X*』(Addison-Wesley Publishing Company, Inc., 1993) の第 4 章を基に記述してあり、著作権所有者の許可を得ています。

ムがサポートするディスプレイ上であれば、プログラムを変更することなく同じ動作や表示を行えます。

DPS の動作

DPS システムにはいくつかのコンポーネントがあり、その中には PostScript インタプリタ、クライアントライブラリ、pswrap トランスレータがあります。クライアントライブラリは、アプリケーションと PostScript インタプリタ間のリンクを行うものです。

DPS 拡張機能を使用する各アプリケーションは、「コンテキスト」を 1 つ生成します。コンテキストは、ウィンドウが見えない画面領域のピックスマップに出力する一種の仮想 PostScript プリンタと考えることができ、専用のスタック、入出力機能、メモリー空間を備えています。PostScript インタプリタはサーバー中で単一のプロセスとして動作しますが、コンテキストが別々になっているため複数のアプリケーションで共有できます。

DPS システムでは 1 つのアプリケーションで複数のコンテキストを使用できますが、1 つのコンテキストで多くの描画領域を扱えるため、通常はコンテキストが1つあればそのアプリケーション内の全描画処理を十分まかなえます。しかし、1 つのクライアント内でコンテキストを 2 つ以上使用する方が好ましい場合もあります。たとえば、Encapsulated PostScript (EPS) ファイルを読み込む場合は、コンテキストを別にする方がよいでしょう。読み込んだ EPS ファイルに PostScript に関するエラーがあった場合、コンテキストを別にしておけばエラー回復処理が簡単になります。

アプリケーションは、クライアントライブラリの手続きを呼び出すことによって、画面上に描画します。この手続きは、PostScript インタプリタに送られて処理される PostScript 言語コードを生成します。DPS システムには、クライアントライブラリの他に pswrap トランスレータが組み込まれています。pswrap は、PostScript 言語のオペレータを使用して「ラップ」という C 言語手続きを生成します。「ラップ」は、アプリケーションプログラムから呼び出すことができます。

PostScript インタプリタは、タイムスライス単位でコンテキスト実行に関連するスケジューリングを行います。このインタプリタはコンテキスト間で切り替えられるので、複数のアプリケーションがインタプリタにアクセスできます。各コンテキストは、PostScript の仮想メモリー空間 (VM) の専用部分にアクセスします。VM には「共有 *VM*」と呼ばれる追加部分もあり、これは全コンテキスト間で共有され、システムフォントや他の共用リソースに使用します。「専用 *VM*」には、各コンテキスト専用のフォントを入れることができます。図 2–1 に DPS のコンポーネントと X との関係を示します。

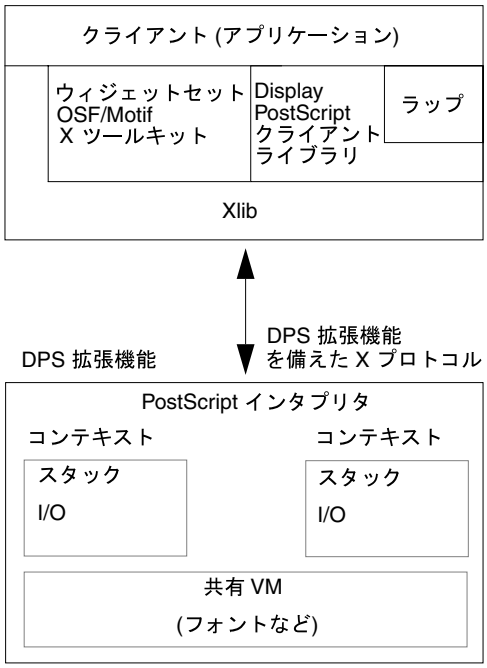


図 2–1　DPS 拡張機能と X

アプリケーションは、次のような手順で DPS システムとやり取りします。

1. アプリケーションは PostScript 実行コンテキストを 1 つ生成し、サーバーとの通信チャネルを 1 つ確立します。
2. アプリケーションは、クライアントライブラリの手続きとラップをコンテキストに送り、その応答を受け取ります。
3. アプリケーションは終了する際、コンテキストを消滅させ、通信チャネルをクローズし、セッション中に使用したリソースを解放します。

コンテキストの構造は、全 DPS プラットフォーム間で共通です。ただし、コンテキストの作成と管理はプラットフォームごとに異なります。コンテキストとそれを処理するルーチンについては、『*Client Library Reference Manual*』と『*Client Library Supplement for X*』を参照してください。Display PostScript の開発者向けユーティリティについては、『*Display PostScript Toolkit for X*』を参照してください。

Solaris サーバーの DPS に対するフォント拡張項目

サーバーでは、DPS システムに対して次のようなフォントを拡張しています。

- F3 のラテン系文字フォントとアジア系文字フォントのサポート
- TrueType フォントのサポート

詳細は、第 4 章を参照してください。

DPS ライブラリ

DPS ライブラリを表 2–1 に示します。これらライブラリの構成要素である `.so` ファイルと `.a` ファイルは、`/usr/openwin/lib` と `/usr/openwin/lib/libp` のディレクトリに入っています。これらのライブラリの詳細については、『*Programming the Display PostScript System with X*』および『*PostScript Language Reference Manual*』を参照してください。

表 2–1 DPS ライブラリ

ライブラリ	説明
`libdps`	DPS クライアントライブラリ
`libdpstk`	DPS ツールキットライブラリ
`libpsres`	PostScript 言語用リソース配置ライブラリ
`libdpstkXm`	DPS Motif ツールキットライブラリ

Adobe NX エージェントのサポート

`libdps` のコンテキスト作成ルーチン (`XDPSCreateSimpleContext` と `XDPSCreateContext`) は、DPS/X の拡張機能に接続できない場合には、DPS NX

エージェントへの接続を試みるようになっています。NX クライアントは、通常、ブート時または X ウィンドウの起動処理中に手動で起動しなければなりません。

Adobe Systems Inc. から入手できる Adobe DPS NX エージェントは、X ウィンドウサーバーや DPS/X クライアントとは別のプロセスです。DPS NX エージェントに接続すると、クライアントの DPS 呼び出しは中断され、標準の X プロトコル要求に変換されます。したがって、DPS の拡張機能がサポートされない X ウィンドウサーバー上でも、DPS クライアントを実行することができます。

DPS のセキュリティ

Solaris 環境では、少なくとも X コンソーシアムの X11R5 サンプルサーバーのセキュリティレベルが得られます。ただし、これを保証するために DPS では、PostScript のファイルオペレータからシステムファイルをアクセスできないようにすると同時に、機密保護コンテキストを生成する機能も提供しています。以下でこの 2 つの機能を説明します。

システムファイルのアクセス

PostScript 言語ではファイル操作によりディスクファイルなどのシステムデバイスにユーザーがアクセスできますが、これがセキュリティ上の重大な問題になります。そのため Solaris 環境ではデフォルト時、PostScript のファイルオペレータでシステムファイルをオープンしたり、アクセスしたりできないようにしています。

アプリケーションの場合は、サーバーではなくクライアントでファイル操作を実行する必要があります。したがって、クライアントには、サーバーに必要な全アクセス特権が必要というわけではありません。PostScript のファイルオペレータでシステムファイルをアクセスしたい場合は、サーバー起動時に `-dpsfileops` オプションを指定してください (`Xsun(1)` のマニュアルページを参照)。このオプションを指定しないでシステムファイルをアクセスすると、PostScript の「`undefinedfilename`」というエラーになります。サーバープロセスはスーパーユーザーによって所有されるので、この問題は CDE や `xdm` 環境では特に重要です。

機密保護コンテキストの生成

DPS のコンテキストは通常グローバルデータを使用できるため、あるコンテキストから別のコンテキストの動作を見ることができます。たとえば、あるコンテキストが作成中のドキュメントを、別のコンテキストから横取りすることもできるでしょう。この節では、Solaris 環境での機密保護コンテキストの生成方法を説明します。

『*PostScript* リファレンスマニュアル、第 2 版』の 7.1.1 節「コンテキストの作成」は、コンテキストが VM を共有するための方法として、次の 3 つを記載しています。

1. 「ローカル VM とグローバル VM は、そのコンテキストに完全に固有である。」この機能はレベル 2 の新しいものですが、これに従って生成したコンテキストを「機密保護コンテキスト」といいます。
2. 「ローカル VM はそのコンテキストに固有であるが、グローバル VM は他のコンテキストと共有されている。」`XDPSCreateContext` や `XDPSCreateSimpleContext` でコンテキストを生成すると、通常こうなります。
3. 「ローカル VM とグローバル VM は、他のコンテキストと共有されている。」`XDPSCreateContext` や `XDPSCreateSimpleContext` で、パラメータの `space` に `NULL` 以外を指定してコンテキストを生成するとこうなります。

   機密保護コンテキストを生成するには、`XDPSCreateSecureContext` を次のように使用してください。

```
XDPSCreateSecureContext

   DPSContext XDPSCreateSecureContext(dpy,    drawable, gc, x, y, eventmask,

   grayramp, ccube, actual,     textProc, errorProc, space)  Display *dpy;

    Drawable drawable;  GC gc;  int x;  int y;  unsigned int eventmask;

    XStandardColormap *grayramp;  XStandardColormap *ccube;  int actual;

    DPSTextProc textProc;  DPSErrorProc errorProc;   DPSSpace

   space;
```

パラメータの意味はすべて `XDPSCreateContext` のものと同じですが、生成されるコンテキストは専用の専用グローバル VM を 1 つ持ちます。パラメータの `space` に `NULL` を指定しない場合は、機密保護コンテキストと一緒に生成する空間を指定しなければなりません。機密保護コンテキストと一緒に生成した空間を、非機密保護コンテキストの生成時に使用することはできません。機密保護コンテキストで非

機密保護空間を指定したり、非機密保護コンテキストで機密保護空間を指定したりすると、アクセスエラーになります。

DPS で内部エラーが発生したときの対策

DPS は実行中に整合性検査を実施します。まれに内部エラーが発生する場合がありますが、そのような場合には、DPS アプリケーションはサーバーに接続することができません。このような状態になった場合は、Solaris 環境を再起動する必要があります。この状態でクライアントが DPS 拡張機能を使用してサーバーに接続しようとすると、次のエラーメッセージが表示されることがあります。

```
XError:

130 Request Major code 129

(Adobe-DPS_Extension)
```

Adobe 社からの情報の入手方法

インターネットまたは UUCP 電子メールを使用して Adobe の公開ファイルにアクセスできる場合は、Adobe 社の一般アクセス用ファイルサーバーを使用して、ソースコード例、AMF (Adobe Metric Font) ファイル、ドキュメンテーション、PPP (PostScript printer description) ファイル、プレスリリースを入手できます。

インターネットを利用する場合は、ファイル転送プログラム (`ftp`) を使用して、`ftp.mv.us.adobe.com` マシンからファイルをダウンロードしてください。入手したファイルの詳細については、`README.first` ファイルを参照してください。電子メールで情報を取得する方法については、『*Programming the Display PostScript System with X*』の「Preface」に記載されている「Public Access File Server」の節を参照してください。

DPS 合成オペレータ

注意 - この節で定義するオペレータは、Display PostScript 言語の拡張機能です。標準 DPS の一部ではないので、どの DPS 実装でも使用できるとは限りません。これらのオペレータに依存するアプリケーションには移植性がなく、これらのオペレータをサポートしないサーバー上では表示できません。

合成とは、Display PostScript システムを OpenStep™ において拡張した機能です。合成により、別々に描画されたイメージを組み合わせて最終的なイメージを生成できます。この拡張機能により、次のようにさまざまなイメージ処理機能が提供されます。

- ある場所から別の場所へというように、PostScript を使用してイメージを単純にコピーする。
- 2 つのイメージを追加して、相互に重なり合って混ぜたように表示する。
- 組み合わせたイメージの一方または両方の透明部分を活用する多数の処理を定義する。イメージが合成されると、一方のイメージが透明なので、他方のイメージの一部を表示できる。

合成機能は、同じウィンドウ内でスクロールしてコピーするとき、またはあるドロアブル内で描画したイメージを取り出して別の領域に転送するときに使用できます。OpenStep アプリケーションでは、一般にイメージはピックスマップに格納され、必要に応じてウィンドウに合成されます。

イメージが部分的に透明な場合は、そのイメージの透明部分によって他方のどの部分が表示されるかを合成により調整できます。合成の各操作において、さまざまな方法で透明部分を利用します。通常の操作では、あるイメージが他のイメージの背景または前景となります。イメージの一部が透明な場合は、不透明な背景の上で合成できます。これにより、上になったイメージにある透明の「穴」を通して見えるようになります。また、操作によっては、あるイメージの透明部分を使用して、合成したイメージの対応部分を「消去」できます。ほとんどの処理では、両方のイメージの透明部分に基づいて合成を計算します。

透明部分を利用する合成では、さまざまな視覚効果を生み出すことができます。一部が透明な均一のグレイ領域を淡彩のように使用すると、一緒に合成したイメージの色を濃くすることができます。一部が透明なグレイのパッチを使用すると、別のイメージに陰を追加できます。2 つのイメージの透明部分を段階的に変更しなが

ら繰り返し合成すると、一方からもう一方へのグラデーションを表示できます。また、固定した背景上で動画を合成できます。

イメージを合成する前に、それを描画しなければなりません。合成時に透明を活用するには、少なくとも 1 つのイメージを透明ペイントで描画する必要があります。

次の PostScript プログラムの抜粋は、合成オペレータの使用方法を示しています。このプログラムは、2 つの単純なイメージを作成して合成します。第 1 のイメージである宛先は、白地に作成される 0.8 のグレイの三角形です。第 2 のソースは、透明の背景上に作成される 0.6 のグレイの三角形です。

```
%
宛先三角形の作成  0.8 setgray  100 100 moveto  100 0 rlineto  0
-100 rlineto  fill  % ソースの背景を透明にする  0
setalpha  0 0 100 100 rectfill  % ソース三角形の描画  1 setalpha  0.6
setgray  0 0 moveto  0 100 rlineto  100 0 rlineto  fill  % 結果の計算
 0 0 100 100 null 100 0 Sover composite
```

8 番目の合成オペレータ、Sover では、ソースピクセルと宛先ピクセルの組み合わせ方が定義されます。この例では、ソースイメージのうち不透明な部分が宛先イメージ上に配置されます。結果的に生成されるイメージは、図 2–2 のようになります。

図 2–2　合成オペレータのサンプルプログラム

オペレータについて

この節では、新しい DPS オペレータについて説明します。この情報は、PostScript のマニュアル『*PostScript Language Reference Manual*』と『*Display PostScript System with X*』で使用されている形式で掲載してあります。

setalpha *coverage* **setalpha**

現在のグラフィックス状態の *coverage* パラメータを *coverage* に設定します。*coverage* は 0 から 1 までの数値で、0 は透明に対応し、1 は不透明な状態に対応し、その間の値は部分的なカバレージに対応します。デフォルトは 1 です。これにより、合成により透けて見える背景の量が設定されます。カバレージ値が 0 より小さければ、*coverage* パラメータは 0 に設定されます。この値が 1 より大きければ、*coverage* パラメータは 1 に設定されます。

カバレージ値は、PostScript のマーキング操作によってペイントされる色に影響します。現在の色は、描画前にアルファ値によってあらかじめ乗算されます。この乗算は、現在の色が RGB 空間に変形された後で実行されます。

エラー **stackunderflow**, **typecheck**

参照 **composite**, **currentalpha**

currentalpha -**currentalpha** *coverage*

現在のグラフィックス状態の *coverage* パラメータを返します。

エラー なし

参照 **composite, setalpha**

composite *srcx srcy width height srcgstate destx desty op* **composite**

2 つのイメージ、ソースと宛先に含まれるピクセルのペア間で、*op* で指定された合成操作を実行します。ソースピクセルは *srcgstate* グラフィックス状態によって参照されるドロアブル内にあり、宛先ピクセルは現在のグラフィックス状態で指定されるドロアブル内にあります。*srcgstate* が NULL の場合、現在のグラフィックス状態が想定されます。

srcx、*srcy*、*width*、*height* で指定された矩形は、ソースイメージを定義します。この矩形の輪郭は、部分的な座標、縮尺、または回転軸によりピクセル境界をまたぐことができます。ソースに含まれるピクセルは、矩形の輪郭に囲まれるすべてのピクセルとなります。

宛先イメージは、ソースと同じサイズ、形状、向きを持っています。*destx* と *desty* は、ソースに対応する宛先の位置イメージを示します。2 つのグラフィックス状態の向きが異なる場合も、イメージの向きは同じで、合成してもイメージは回転されません。

両方のイメージは、それぞれのドロアブルの枠の矩形にクリップされます。宛先イメージは、現在のグラフィックス状態のクリッピングパスにさらにクリップされます。合成操作の結果によって宛先イメージが置換されます。

op は合成操作を指定します。操作後の各宛先イメージのピクセル (アルファ値) の色、dst′ (dstA′) は、次のように与えられます。

```
 dst′ = src *
Fs(srcA, dstA, op) + dst * Fd(srcA, dstA, op)

 dstA′ = srcA *
Fs(srcA, dstA, op) + dstA * Fs(srcA, dstA, op)
```

この場合、`src` と `srcA` はソースの色とアルファ値、`dst` と `dstA` は宛先の色とアルファ値、`Fs` と `Fd` は 表 2–2 に示す関数です。

表 2–2 は合成 *op* の選択肢を示します。各操作結果については、図 2–3 を参照してください。

エラー **`rangecheck, stackunderflow, typecheck`**

参照 **`compositerect, setalpha, setgray, sethsbcolor, setrgbcolor`**

表 2–2 合成式の係数

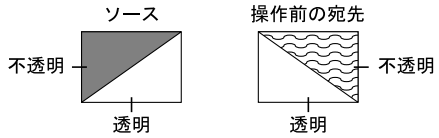
Op	Fs	Fd
`Clear`	0	0
`Copy`	1	0
`Sover`	1	1 - srcA
`Sin`	dstA	0
`Sout`	1 - dstA	0
`Satop`	dstA	1 - srcA

表 2–2　合成式の係数　続く

Op	Fs	Fd
`Dover`	1 - dstA	1
`Din`	0	srcA
`Dout`	0	1 - srcA
`Datop`	1 - dstA	srcA
`Xor`	1 - dstA	1 - srcA
`PlusD`[1]	N/A	N/A
`PlusL`[2]	1	1

1. PlusD は一般の式には準拠していません。この式は dst′=(1-dst)+(1-src) です。この結果が 0 (黒) より小さい場合、結果は 0 となります。
2. PlusL の場合、加算によって飽和状態 (1) になります。つまり、(src+dst) >white) の場合、結果は白になります。

図 2–3 は、合成操作の結果を示します。



操作	操作後の宛先
Copy	ソースイメージ
Clear	透明
PlusD	カラー値が限界の 0 に近づいている箇所の、ソースイメージと宛先イメージとの和
PlusL	カラー値が限界の 1 に近づいている箇所の、ソースイメージと宛先イメージとの和
Sover	ソースイメージが不透明な箇所は、ソースイメージ、それ以外は宛先イメージ
Dover	宛先イメージが不透明な箇所は、宛先イメージ、それ以外はソースイメージ
Sin	両方のイメージが不透明な箇所は、ソースイメージ、それ以外は透明
Din	両方のイメージが不透明な箇所は、宛先イメージ、それ以外は透明
Sout	ソースイメージが不透明で宛先イメージが透明な箇所は、ソースイメージ、それ以外は透明
Dout	宛先イメージが不透明でソースイメージが透明な箇所は、宛先イメージ、それ以外は透明
Satop	両方のイメージが不透明な箇所のソースイメージ、宛先イメージが不透明でソースイメージが透明な箇所の宛先イメージ、それ以外は透明
Datop	両方のイメージが不透明な箇所の宛先イメージ、ソースイメージが不透明で宛先イメージが透明な箇所のソースイメージ、それ以外は透明
Xor	ソースイメージが不透明で宛先イメージが透明な箇所のソースイメージ、宛先イメージが不透明でソースイメージが透明な箇所の宛先イメージ、それ以外は透明

図 2–3　合成操作の結果

compositerect *destx desty width height op* **compositerect -**

一般に、このオペレータは合成オペレータと同じですが、現実のソースイメージはありません。宛先イメージは現在のグラフィックス状態になっています。*destx*、*desty*、*width*、*height* は、そのグラフィックス状態の現在の座標系で宛

先イメージを記述します。宛先イメージに及ぼす効果は、これらがグラフィックス状態の現在の色とカバレージパラメータで指定された色とカバレージで塗りつぶされたソースイメージと同じです。*op* には、合成オペレータの *op* オペランドと同じ意味がありますが、さらに Highlight 操作を実行できます。

Highlight により、ピクセルのカバレージ値に関係なく、宛先矩形内の白の各ピクセルがライトグレイになり、ライトグレイの各ピクセルが白になります。ライトグレイは 2/3 として定義されます。これと同じ操作を繰り返すと、効果が逆転します (モノクロディスプレイ上では、Highlight によって各ピクセルが反転され、白は黒に、黒は白になります)。

注 - Highlight 操作では、ピクセルのカバレージ構成要素の値は変化しません。ピクセルの色とカバレージの組み合わせを引き続き有効にしておくには、Highlight 操作を一時的なものにして、さらに合成する前に元に戻す必要があります。

---

compositerect の場合、宛先イメージに含まれるピクセルは、指定された矩形の輪郭で囲まれるピクセルです。宛先イメージは、現在のグラフィックス状態のウィンドウの枠の矩形とクリッピングパスにクリップされます。

エラー **rangecheck, stackunderflow, typecheck**

参照 **composite, setalpha, setgray, sethsbcolor, setrbgcolor**

dissolve *srcx srcy width height srcgstate destx desty delta* **dissolve -**

この操作には、ソースイメージと宛先イメージを混ぜる効果があります。最初の 7 つの引数では、合成するソースピクセルと宛先ピクセルを選択します。正確な混合割合は *delta* で指定されます。*delta* は、0.0 から 1.0 までの浮動小数点数です。混合されるイメージは次のようになります。

```
delta * source + (1-delta) * destination
```

srcgstate が NULL の場合、現在のグラフィックス状態が想定されます。

エラー **stackunderflow, typecheck**

参照 **composite**

合成 *op* の値は、PostScript の systemdict 内のアプリケーションに使用できます。定義は次のとおりです。

/Clear 0 def

/Copy 1 def

/Sover 2 def

/Sin 3 def

/Sout 4 def

/Satop 5 def

/Dover 6 def

/Din 7 def

/Dout 8 def

/Datop 9 def

/Xor 10 def

/PlusD 11 def

/Highlight 12 def

/PlusL 13 def

実装に関する注と制限

部分的に透明 (半透明) なアルファ

完全に不透明でない (1) か完全に透明でない (0) アルファ値は、慎重に使用する必要があります。半透明による合成操作では、DPS のカラーキューブとグレイランプ内で多数の色を使用できる場合にのみ、最高のイメージ品質が得られます。つまり、イメージ品質は、24 ビット TrueColor または 8 ビット StaticGray ビジュアルのときに最高で、8 ビットの PseudoColor ビジュアルでは貧弱になります。また、半透明ピクセルの場合は、余分な計算が必要になるので、合成操作の性能は大幅に低下します。

インデックスカラービジュアル

Highlight *op* で最高の結果を得るには、DPS のコンテキストのグレイランプ内のカラー数を次のようにする必要があります。

```
fract(((float) numgrays - 1)* 2. / 3.) == 0
```

つまり、(numgrays = 4, 7, 6, 8, 16, ...) となります。これにより、カラー 2/3 グレイはハーフトーンになりません。

DPS のカラーキューブとグレイランプ内で一般に使用可能なカラー数は限られているので、完全に不透明でない (1) か、完全に透明でない (0) アルファ値を持つイメージでは、最高のイメージ品質が得られません。

合成操作は、gstate で指定されたグレイランプまたはカラーキューブ内のピクセル値についてのみ定義されます。カラーキューブとグレイランプに含まれない値を持つピクセルを合成しても、期待した結果が得られないことがあります。

モノクロディスプレイ

アルファ値が 0 または 1 に等しくない 1 ビットのドロアブルの場合、合成操作の結果は不定です。

op Highlight により、1 ビットのドロアブル上のピクセルの色が反転されます。

X 描画操作との対話

不透明ではないアルファを使用して描画されたドロアブルには、アルファチャネルという余分のピクセルが関連付けられています。X Window システムの操作はアルファチャネルには影響しませんが、次の例外があります。

- アルファチャネルを持つウィンドウが露出されているときに、そのウィンドウの X 背景が定義されている場合 (background != None)、背景がペイントされると露出したピクセルのアルファ構成要素は alpha = 1 でペイントされる。
- ウィンドウのサイズが変更されると、アルファチャネル記憶域のサイズが変更される。

アルファチャネルの削除

erasepage オペレータは、グラフィックス状態の現在のドロアブルを不透明な白でペイントします。したがって、ドロアブル内のすべてのピクセルのアルファ値は 1 に等しくなり、アルファチャネル記憶域は削除されます。

デプスが等しくないドロアブル

デプスが等しくないドロアブルの合成は不定です。

第 3 章

Solaris X サーバー上のビジュアル

この章では、Solaris X サーバー上の X ウィンドウのビジュアルを説明します。この章で説明する内容は次のとおりです。

- デフォルトのビジュアル
- 複数デプスデバイス上のビジュアル
- ガンマ補正されたビジュアル
- ビジュアルによるウィンドウプログラミングのヒント

ビジュアル

1 つの表示デバイスで、1 つ以上の表示フォーマットをサポートできます。X ウィンドウシステムでは、ウィンドウサーバーがサポートする表示フォーマットを「ビジュアル」という形式でクライアントアプリケーションに通知します。ビジュアルとは、表示デバイスがサポートする表示フォーマットを記述したデータ構造体です。

X11 クライアントがウィンドウを作成する場合、そのウィンドウのビジュアルを指定します。ビジュアルは、ウィンドウ内の各ピクセルの表示特性を記述します。つまり、ウィンドウのビジュアルは表示デバイスのビデオハードウェアに対して、ウィンドウのピクセル値を解釈する方法を指示します。

システム内に構成されたディスプレイごとに、1 つずつ X11 スクリーンがあります。スクリーンごとに、サポートされるビジュアルのリストがサーバーによってクライアントアプリケーションにエクスポートされます。このビジュアルリストは、

どの表示フォーマットをウィンドウの作成に使用できるかをクライアントアプリケーションに伝えます。

サーバーによって表示スクリーン用にエクスポートされるビジュアルは一定ではなく、スクリーンのデバイスハンドラに依存します。ビジュアルのエクスポートはデバイスハンドラに制御されるので、クライアントアプリケーションは、1、8、24 ビットのように従来のデプスとは異なるデプスを持つビジュアルなど、さまざまなビジュアルを処理できるように準備しなければなりません。デプスが 4、16、および奇数のビジュアルはエクスポートできない可能性があり、クライアントでそれを処理するように準備しなければなりません。

クライアントアプリケーションでは、Xlib ルーチン `XGetVisualInfo(3)` または `XMatchVisualInfo(3)` を呼び出して、スクリーン用にサポートされるビジュアルのリストを照会したり、ユーティリティ `xdpyinfo(1)` を使用してサポートされるビジュアルのリストを照会したりできます。X11 の色とビジュアルの概要については、このマニュアルの「はじめに」に掲載されている X11 のマニュアルを参照してください。

デフォルトビジュアル

X11 スクリーンごとに、スクリーン用にエクスポートされるビジュアルのいずれかがデフォルトビジュアルとして指定されています。デフォルトビジュアルとは、スクリーンのルートウィンドウに割り当てられたビジュアルのことで、ほとんどのアプリケーションではこのビジュアルを使用してウィンドウを作成します。クライアントアプリケーションが起動すると、アプリケーションで別のビジュアルを指定しない限り、そのウィンドウにはデフォルトビジュアルが割り当てられます。

「組み込みデフォルトビジュアル」とは、Solaris X サーバー内でハードコード化されたビジュアルです。スクリーンごとに、そのスクリーンの表示デバイスの特性によって異なるデフォルトビジュアルがあります。`openwin(1)` の実行時に別のデフォルトビジュアルを指定しない限り、これがデフォルトビジュアルとなります。

ウィンドウサーバーが X11 接続ブロック内で返すデフォルトビジュアルを変更したいことがあります。デフォルトビジュアルでは実行できないクライアントプログラムを特定のビジュアルで実行させる場合などです。たとえば、TrueColor ビジュアルがデフォルトビジュアルとなっている 24 ビットデバイス上では、24 ビットカラーで実行できないアプリケーションを PseudoColor ビジュアルで実行させることができます。

複数デプスデバイスの開発担当者にとっては、デフォルトビジュアルを変更すると、アプリケーションの機能をさまざまな構成でテストできるので便利です。デフォルトビジュアルを変更する方法についての詳細は、xsun(1) のマニュアルページを参照してください。サーバーによってエクスポートされるデフォルトビジュアルとサポートされるビジュアルのリストは、`XGetVisualInfo(3)` を使用して X11 から検証できます。

複数デプスデバイスのビジュアル

Solaris X サーバーでは、同時に複数ピクセルデプスのウィンドウを表示できるデバイスがサポートされます。これらのデバイスを、「複数デプス」デバイスといいます。このようなデバイスのほとんどがデプスごとにビットプレーングループを持っているため、しばしば「複数プレーングループ *(MPG)*」デバイスと呼ばれます。

デプスごとに、1 つまたは複数のビジュアルがエクスポートされることがあります。ほとんどの MPG デバイスでは、エクスポートされた任意のビジュアルを使用してウィンドウを作成できます。TrueColor ビジュアルを推奨するアプリケーションの場合、開発担当者は TrueColor ビジュアルが使用可能かどうかを判断する必要があります。これは、PseudoColor がデフォルトビジュアルの場合でも、TrueColor ビジュアルを使用できることがあるからです。

ビジュアルに関するプログラミング上のヒント

この節では、さまざまなビジュアルをサポートするデバイスを対象に X11 アプリケーションのプログラムを作成する際、発生する問題について説明します。

デフォルトビジュアルに関する仮定

X11 クライアントのプログラミングにおいて多い間違いは、デフォルトビジュアルにインデックスクラス (たとえば PseudoColor または StaticColor) があると仮定することです。しかし、デフォルトビジュアルがデバイスによっては 24 ビット TrueColor である可能性もあります。これらのデバイス上で実行する必要があるクライアントについては、そのデフォルトビジュアルの種類に対処できるように準備しておかなければなりません。

よくある誤りを以下に示します。

- デフォルトデプスが 8 であると仮定する。

- カラーマップが書き込み可能と仮定する。
- XGetVisualInfo を使用して適切なビジュアルを探索せず、不適切なデフォルトビジュアルを使用する。

デバイスがクライアントから要求されたビジュアルをサポートしない場合は、次のエラーメッセージが戻されます。このエラーメッセージのうち、# は要求されたデプス番号を表し、*n* は要求された表示デバイスを表します。表 A–1 に示すサポートされるビジュアルとデバイスの組み合わせで、このメッセージが戻される場合は、インストールに問題があります。

```
Error:

cannot provide a default depth #
for device

/dev/fbs/n
```

一般に、各種のデフォルトビジュアルが存在する中でクライアントの移植性を高めるためには、部分的な変更が必要とされます。

境界ピクセルの設定

デフォルトビジュアル以外のビジュアルを使用してウィンドウを作成するときに、アプリケーションがウィンドウ属性構造内で border_pixel 値を設定しなければ、BadMatch エラーが発生します。これは一般的なプログラミングエラーで、デバッグが困難な場合があります。境界ピクセルの設定についての詳細は、XCreateWindow のマニュアルページを参照してください。

注 - グラフィックスとダブルバッファリングの性能が十分に得られない場合 (加速不足など) は、OpenWindows が root としてインストールされていない可能性があります。

ガンマ補正ビジュアル

ビジュアルの線形性属性に応じて、表示されるカラーの彩度応答は異なります。ブラウン管 (CRT) モニターでは、表示されるカラーは実際に要求されたカラーよりも暗くなります。このような暗化現象の原因は、モニターの物理的構造にあります。

デバイスによってはこの暗化現象を補正するビジュアルをサポートしています。これを「ガンマ補正」と呼びます。

ガンマ補正は、フレームバッファから出る色を変更し、モニターの応答を反転させることによって行われます。ガンマ補正ビジュアル全体の彩度は直線的に変化するので、ガンマ補正ビジュアルのことを線形ビジュアルといいます。ガンマ補正されていないビジュアルを非線形ビジュアルといいます。

線形性は、X11 ビジュアルの標準属性ではありません。しかし、一部のアプリケーションでは、視覚的な悪影響を避けるために線形ビジュアルが必要になります。たとえば、平滑化された線分を使用するグラフィックスアプリケーションでは、線形ビジュアルを使用しなければ、好ましくない「ローピング」アーティファクトが生じることがあります。このようなアプリケーションは線形アプリケーションと呼ばれます。カラーの最適な表示に非線形ビジュアルを必要とするアプリケーションは、非線形アプリケーションと呼ばれます。ほとんどの X11 アプリケーションは、非線形アプリケーションです。

ほとんどのデバイス上のデフォルトビジュアルは非線形です。したがって、線形アプリケーションはデフォルトビジュアルを使用せず、線形ビジュアルを常に明示的に探索しなければなりません。同様に、非線形アプリケーションの場合も、非線形ビジュアルを明示的に検索するとよいでしょう。これは一般にほとんどのデバイスのデフォルトなので不可欠ではありませんが、望ましい方針ということができます。

ビジュアルが線形かどうかを判断するには、アプリケーションはインタフェース `XSolarisGetVisualGamma(3)` を使用できます。ガンマ補正についての詳細は、Foley and Van Dam 著『*Fundamentals of Computer Graphics*』を参照してください。

ビジュアル選択の代替方法

ビジュアルを選択する方法としては、各種のビジュアル構成を処理できるようなアプリケーションを作成することをお勧めします。GX のような一部のデバイスでは、線形ビジュアルがサポートされません。また、単一の線形 24 ビット TrueColor ビジュアルだけをサポートするデバイスがあります。さらに、線形ビジュアルと非線形ビジュアルが同時にサポートされるデバイスもあります。一般に、移植性のあるアプリケーションを作成する最も賢明な方法は、これらすべての構成を細かく操作することです。希望の線形性を持つビジュアルが見つからない場合は、警告メッセージを出力するようにします。あるいは、線形アプリケーションで線形ビジュアルが見つからない場合は、便利なトリックとして、X11 に与えられたカラーをアプリケーション内で手作業で暗くする方法もあります。これは独自のガンマ補正を実

行するのと同じことです。カラーをどの程度暗くするか
は、XSolarisGetVisualGamma で戻されるガンマ値から判断できます。

注 - XSolarisGetVisualGamma は、Solaris の「公開」インタフェースで、全面的にサポートされます。将来は、カラー管理システムでもこの機能が提供される可能性があります。そうすれば、この情報を取得する上で望ましい方法となります。しかし、それまでは XSolarisGetVisualGamma を使用してガンマ値を調べてください。このカラー管理システムが導入されると、XSolarisGetVisualGamma を使用するアプリケーションは修正しなくても引き続き動作し、実際にはカラー管理システムによって精度が高くなるという利点が得られます。

第 4 章

フォントのサポート

この章では、Solaris X サーバーでのフォントのサポートについて説明します。この章で説明する内容は以下のとおりです。

- X フォントサーバー
- 使用できるフォントフォーマット
- アウトラインフォントとビットマップフォント
- フォントの位置

Solaris X サーバーでのフォントのサポート

Solaris X Window System は、X11 サーバーと Display PostScript (DPS) 拡張機能にフォントのサポートを提供します。さまざまなベンダ製のフォントフォーマットを使用して、英語やアジア系言語などの外国語のテキストを表示できます。シンボルフォントを使用すると、数学の方程式も表示できます。Solaris 環境では、欧文テキスト用のラテン系フォントを 55 種類、シンボルフォントを 2 種類用意しています。Solaris に付属のフォント管理の GUI ツールまたはコマンドラインツールを使用して、その他のフォントもシステムに追加できます。

X フォントサーバー

Solaris X サーバーは X フォントサーバー `xfs` のクライアントにできます。X フォントサーバーは X サーバーにフォントを提供します。Solaris X フォントサーバー

は、標準の X フォントサーバーと同じフォントに加えて、Sun の TrueType フォントもサポートします。Sun 独自の F3 フォントフォーマットはサポートしていません。Type1 フォントは、X コンソーシアムに寄贈された Type1 インタプリタによりサポートされます。

`xfs` は手動または自動で起動できます。詳細は `xfs(1)` のマニュアルページを参照してください。

使用できるフォントフォーマット

ベンダが異なればフォントのフォーマットも異なります。Solaris 環境でサポートするフォントフォーマット、ベンダ、対応するファイルタイプを、表 4–1 と表 4–2 に示します。表 4–1 はアウトラインフォント、表 4–2 はビットマップフォントをリストします。

表 4–1 アウトラインフォントフォーマット

フォントフォーマット	ベンダ	ファイルタイプ
TrueType	その他のメーカ	`.ttf`
Type1 (ASCII)	Adobe 社、その他のメーカ	`.pfa`
Type1 (バイナリ)	Adobe 社、その他のメーカ	`.pfb`
Type3	Adobe 社、その他のメーカ	`.ps`
Speedo	Bitstream 社	`.spd`
F3	SunSoft	`.f3b`

表 4–2 ビットマップフォントフォーマット

フォントフォーマット	ベンダ	ファイルタイプ
可搬コンパイル済形式	MIT	`.pcf`
ビットマップ配布形式	Adobe 社	`.bdf`

表 4–2 ビットマップフォントフォーマット 続く

フォントフォーマット	ベンダ	ファイルタイプ
ビックエンディアン可搬形式	Adobe 社 (sparc)	`.bepf`
リトルエンディアン可搬形式	Adobe 社 (IA および ppc)	`.lepf`

Solaris サーバーによって提供されるフォントは、`/usr/openwin/lib/X11/fonts` ディレクトリに入っています。ディレクトリ構造の詳細については、後述の 61ページの「フォントの探索」を参照してください。

Solaris 環境では、DPS からでも X11 フォントの大部分を使用できるようになっています (表 4–3 を参照)。ただし、DPS でサポートするフォントは X11 とは少し異なっています。

表 4–3 利用できるフォントファイル

フォント名	**X11** での使用	**DPS** での使用
TrueType	可	可
Type1 アウトラインフォント (ASCII)	可	可
Type1 アウトラインフォント (バイナリ)	可	可
Type3	可	可
Speedo	可	不可
F3	可	可
可搬コンパイル済形式	可	可
ビットマップ配付形式	可	不可
ビックエンディアン可搬形式	不可	可
リトルエンディアン可搬形式	不可	可

オプションのフォントパッケージ

エンドユーザーアプリケーションが必要とするフォントはエンドユーザークラスタでインストールされます。しかし、一部の特殊なアプリケーションではデベロッパクラスタのフォントが必要になります。このようなアプリケーションに対応するために追加するパッケージは SUNWxwoft パッケージです。デベロッパクラスタ全体をインストールする必要はありません。

関連ファイル

Solaris 環境では、次のような拡張子のファイルを提供します。これらの内容を編集しないでください。

- `.afm` カーニング情報を入手するためにクライアントが読む Adobe フォントメトリックファイル
- `.map` エンコーディング用に X11 と DPS が読む F3 ファイル
- `.trans` 複合フォント作成用に DPS が読む F3 ファイル
- `.ps` 複合フォントと PostScript リソースを作成するための PostScript ファイル
- `.enc` X11 と DPS が使用するエンコーディング用ファイル
- `.upr` Display PostScript リソースファイル
- `.ttmap` TrueType フォント用のエンコーディングファイル

アウトラインフォントとビットマップフォント

Solaris のフォントの表現方法には、「アウトライン」フォントと「ビットマップ」フォントの 2 種類があります。アウトラインフォントから文字を表示する際、サーバーは文字の輪郭線だけについて拡大・縮小と回転処理を行います。こうして形を整えられた輪郭線は、ピクセル形式 (ビットマップ) で「描画」され、画面上に表示されます。このビットマップは、再使用のためにグリフキャッシュにも格納されます。

頻繁に使用されるのは一部のフォントサイズであるため、そのビットマップはプリレンダリング済みビットマップ形式の別のファイルにも格納されます。これによりサーバーは、拡大・縮小やレンダリングの処理を行う手間が省けます。ただし、結果として得られるビットマップフォントはサイズと方向が一意に決まってしまうた

め、「手で調整」することによって見映えよく読みやすくするフォントもあります。これらのビットマップもグリフキャッシュに格納されます。推奨するビットマップフォーマットは、可搬コンパイル済形式 (.pcf) です。

/usr/openwin/bin というディレクトリには、さまざまなビットマップフォーマット間だけでなく、アウトラインフォントとビットマップフォント間の変換も行うツールとして次のものが入っています。詳細については、それぞれのマニュアルページを参照してください。

- makebdf – F3 アウトラインフォントファイル (.f3b) からビットマップ配付形式フォーマットのファイル (.bdf) を生成する。
- bdftopcf – .bdf フォーマットから可搬コンパイル済形式 (.pcf) にフォントを変換する。

表 4–4 に示すように、ビットマップフォントのファイルフォーマットの多くは、アーキテクチャに依存したバイナリファイルです。これらは、アーキテクチャの異なるマシン間 (SPARC と IA 間など) では共有できません。

表 4–4 ビットマップフォントのフォーマット

フォントフォーマット	バイナリ形式	アーキテクチャに依存
ビットマップ配付形式	いいえ	いいえ
可搬コンパイル済形式	はい	いいえ
リトルエンディアン可搬形式	はい	はい (IA および ppc)
ビッグエンディアン可搬形式	はい	はい (SPARC)

Solaris 環境には圧縮形式の .pcf ファイル (拡張子は .pcf.Z) があり、必要に応じて展開できます。システムにフォントを追加する際、フォントファイルは圧縮してもしなくてもかまいません。フォントをより高速で表示したい場合は、非圧縮形式のファイルを使用してください。ディスク領域をより多く確保するには、圧縮ファイルを使用してください。詳細については、compress(1) のマニュアルページを参照してください。

アウトラインからビットマップへのフォントの置き換え

アウトラインフォントによっては、フォントサイズが合えば Solaris 環境がビットマップフォントに自動的に置き換えるものもあります。これにより性能が向上し、場合によってはテキストも美しく読みやすくなります。1 つのアウトラインフォントに対して、置き換えが発生するサイズにはいくつかあります。

フォントの置き換えが発生する条件

現在の DPS では、F3 アウトラインフォント、Type1、および TrueType フォントは .pcf ビットマップフォーマットで置き換えられます。置き換えが発生するのは、正立で、要求された 1 ピクセルのサイズが .pcf フォントサイズの 1 ピクセルの半分以下であり、.pcf フォントが .upr (PostScript リソース) ファイル中のリソースになっている場合です。上記のすべてのスケーラブルフォントは .pcf フォーマットで置き換えられます。

DPS で TrueType および F3 フォントを使用する場合

/FontType が TrueType については 42、F3 フォントについては 7 を返すことを除いて、TrueType および F3 フォントは Type1 フォントと同様に動作します。たとえば、次の PostScript コードは、フォントの種類にかかわらず同じように動作します。

```
/Helvetica
findfont 50 scalefont setfont 10 10 moveto (ABC)

show
```

しかし、次のコードを実行すると、TrueType フォントでは 42、F3 フォントでは 7、Type1 フォントでは 1 が戻されます。

```
currentfont
/FontType get ==
```

戻されるフォントの種類は、DPS の現在の内部リソースパスによって決まります。

フォントの探索

デフォルト時 Solaris サーバーは、/usr/openwin/lib/X11/fonts というディレクトリに入っているディレクトリの中からフォントを探索します。フォントのディレクトリ構造全体については、次の表 4–5 を参照してください。ディレクトリ名には /usr/openwin/lib/X11/fonts というパス名がつきます。

表 4–5 フォントのディレクトリ構造

ディレクトリ	サブディレクトリ	ファイル拡張子	内容
/TrueType		.ttf	TrueType フォント
/TrueType	/ttmap	.ttmap	TrueType 文字セット仕様
/TTbitmaps		.pcf	ビットマップフォント
/100dpi		.pcf	ビットマップフォント
/75dpi		.pcf	ビットマップフォント
/F3	/afm	.f3b	F3 フォーマットのスケーラブルフォント
	/map	.map	F3 文字セット仕様
/F3bitmaps		.pcf	ビットマップフォント
/Speedo		.spd	Bitstream Speedo フォーマットのアウトラインフォント
/Type1		.pfa, .pfb	Type1 スケーラブルフォント
	/afm	.afm	Adobe フォントメトリック
	/outline	.pfa, .pfb	Type1 スケーラブルフォント
	/prebuilt	.bepf, .lepf	SPARC Solaris と IA 用ビットマップ

表 4–5　フォントのディレクトリ構造　続く

ディレクトリ	サブディレクトリ	ファイル拡張子	内容
/Xt+		.pcf	ビットマップフォント
/Type3		.ps	PostScript アウトラインフォント
/encodings		.enc	エンコーディング情報
/misc		.pcf	ビットマップフォント

X11 でのデフォルトフォントパスの変更

X11 でのデフォルトフォントパスは、次のようになっています。

```
/usr/openwin/lib/X11/fonts/F3, /usr/openwin/lib/X11/fonts/
F3bitmaps, /usr/openwin/lib/X11/fonts/Type1, /usr/openwin/lib/
X11/fonts/Speedo, /usr/openwin/lib/X11/fonts/misc, /usr/openwin/
lib/X11/fonts/75dpi, /usr/openwin/lib/X11/fonts/100dpi
```

ディレクトリパスは絶対パスでなければなりません。

デフォルトのフォントパスを変更するには、Solaris に付属のフォント管理 GUI ツールまたはコマンドラインツールを使用します。フォント管理ツールについての詳細は、『フォントの管理』を参照してください。

フォントのインストールおよび管理

ワークステーションまたは NeWSprint™ プリンタのフォントをインストール、削除、表示したり、フォントパスやフォント属性を編集するには、Solaris に付属の管理 GUI ツールまたはコマンドラインツールを使用します。フォント管理ツールについての詳細は、『フォントの管理』を参照してください。

第 5 章

サーバーオーバーレイウィンドウ

この章では、以下の項目について説明します。

- サーバーオーバーレイと Solaris 透明オーバーレイ
- オーバーレイ実装時のヒント
- サーバーオーバーレイの説明

サーバーオーバーレイと透明オーバーレイ

オーバーレイウィンドウに透明なピクセルを描画する場合、2 つの API を使用できます。透明オーバーレイ拡張機能は Sun 独自の方法で X Window System にオーバーレイ機能を提供します。透明オーバーレイは、ハードウェアのオーバーレイサポートがない場合でもオーバーレイ機能を提供できます。ハードウェアがサポートしている場合には、サーバーオーバーレイというもう 1 つの方法を使用できます。

透明オーバーレイ拡張機能は完全な X 拡張機能で、透過性の効果を得るために拡張機能を呼び出す必要があります。この方法は確実であり、ハードウェアが実際にオーバーレイをサポートしていない場合でも、ほとんどのシステムで透過性をエミュレートできます。しかし、ハードウェアでサポートされていない場合には、透明なウィンドウの操作は非常に遅くなります。

サーバーオーバーレイは X 拡張機能ではなく、代わりに API が X クライアントに対して、オーバーレイであるビジュアルと、透過性のために使用するピクセル値を判断するための方法を提供します。この API はハードウェアサポートを必要とします。

透明オーバーレイ拡張機能およびサーバーオーバーレイは同じスクリーン上で使用できますが、同じウィンドウ内で使用しないでください。結果は未定義となります。サーバーオーバーレイ専用に設計されたビジュアル内で透明オーバーレイウィンドウを作成しようとすると、BadMatch が発生します。透明オーバーレイでは、適切な手順に従ってパートナオーバーレイビジュアルを特定することによりこれを回避できます。第 6 章を参照してください。

オーバーレイのプログラミングのヒント

以下の情報は、使用するモデルとスタッキングの処理方法を決めるときに役立ちます。

親子モデル

すべての透過性およびオーバーレイの設計において、アンダーレイ (下層) が親、オーバーレイが子という単純なモデルを使用することをお勧めします。アンダーレイウィンドウを最初に作成した後、アンダーレイの子としてオーバーレイを作成します。オーバーレイはアンダーレイの唯一の子です。これにより、X サーバーの多くの奇妙な状況が回避され、アンダーレイとオーバーレイの間に誤ってウィンドウが挿入されることがなくなります。

`Xlib` を使用したり、これらの呼び出しについて独自の `XCreateWindow` をプログラミングしたりする場合、親と異なるビジュアルを持つウィンドウを作成するときにはクライアントはより多くの情報を提供しなければなりません。ビジュアルがデフォルトのビジュアルでない場合、カラーマップを指定するか、カラーマップが同じ場合は親のビジュアルのカラーマップを子に割り当てなければなりません。デプスが異なる場合は、BorderPixel または BorderPixmap を指定しなければなりません。そうしないと、ウィンドウ作成の結果として BadMatch が返されることがあります。

カラーマップの等価性については、『*X Server Device Developer's Guide*』を参照してください。

スタッキング

ウィンドウを手前に配置する場合、そのウィンドウがオーバーレイウィンドウであるかどうかに関係なく、スタックの一番上に配置されます。ウィンドウを後ろに配置する場合、そのウィンドウがオーバーレイウィンドウであるかどうかに関係なく、スタックの一番下に配置されます。

これにより、オーバーレイウィンドウがアンダーレイウィンドウの下になるという混乱が生じます。実際に、このようなことは頻繁に発生します。これは X サーバーが単純なスタッキングポリシーを使用するためで、ソフトウェアでクリップしなければならない場合でも、あらゆる処理を行ってオーバーレイウィンドウを他のウィンドウの下に表示します。

アンダーレイが親、オーバーレイが子というモデルを使用することにより、この問題を解決できます。これにより、アンダーレイとオーバーレイのペアは、親ウィンドウのアプリケーションとして処理され、一緒に手前および後ろに移動されます。

サーバーオーバーレイ

サーバーオーバーレイ API により、アプリケーションは簡単にオーバーレイビジュアルと対応する透明なピクセル値を見つけることができます。オーバーレイビジュアルはオーバーレイウィンドウを作成するために使用され、透明なピクセルは下にあるアンダーレイが見えるようにするためにクライアントが使用する特殊なピクセル値です。このピクセル値は、すべての描画操作での一般的な前景や背景、またはオーバーレイウィンドウの背景で使用されます。

サーバーオーバーレイ API は、ルートウィンドウの SERVER_OVERLAY_VISUALS プロパティに以下の情報が含まれていることを指定しています。サーバーにより返される情報のサイズにより、返されるこの構造体のインスタンスの数 (リストされた各ビジュアルについて 1 つのインスタンス) が決まります。

```
typedef struct {
 unsigned int visualid;   unsigned int
trans_type;   unsigned int
value;   unsigned int
layer;   } ServerOverlaysInfoRec;
```

visualid	X サーバーにより参照されるビジュアル ID。通常、`XGetVisualInfo` によりクライアントに返される。
trans_type	透過性のタイプ。0 は None、1 は Transparent Pixel、2 は Transparent Mask。
value	透明なピクセル値またはマスク値。
layer	透過性の効果に関するビジュアルの相対的なハードウェアレイヤ。

trans_type の値は、仕様では一般的ではない他の透過性のタイプがある、という前提で存在しています。透明なピクセルが使用できない場合 *trans_type* はゼロになりますが、これは X サーバーが、通常のウィンドウとは異なるプレーングループのセットに存在することを、知らせようとするためです。

通常のウィンドウの場合 layer は普通ゼロですが、実際には相対的な数値で、数値の大きいプレーングループは数値の小さいプレーングループよりも上であることを示します。負の数値の場合もあります。

`SERVER_OBERLAY_VISUALS` プロパティにリストされていないビジュアルは、レイヤが 0、透過性が none であるとみなされます。これらのデフォルト値は、サーバーオーバーレイ操作にのみ適用されます。

透明なピクセルは次のレイヤの最初のウィンドウに通過して表示されます。レイヤはスタック順序に影響せず、透過性の効果にのみ適用されます。対象となるアンダーレイの直接かつ唯一の子としてオーバーレイを使用することをお勧めします。これにより、最適のパフォーマンスが得られ、混乱が少なくなります。

サーバーオーバーレイのサポートはデバイスに依存し、完全なハードウェア移植や部分的なソフトウェアエミュレーション、またはソフトウェアとハードウェアの組み合わせの場合があります。

サーバーオーバーレイは、『*The X Journal*』July/August 1993 号の Mark J. Kilgurad による「Programming X Overlay Window」で定義されています。

第 6 章

透明オーバーレイウィンドウ

この章では、Solaris OpenWindows 環境で透明オーバーレイウィンドウ機能を提供する、透明オーバーレイ拡張機能アプリケーションプログラミングインタフェース (API) について説明します。

- オーバーレイウィンドウと標準 X ウィンドウの違い
- オーバーレイウィンドウの作成方法と描画方法
- 透明オーバーレイウィンドウを使用するアプリケーションについて、さまざまなデバイスへの移植性を保証する方法

注 - ハードウェアでサポートされている場合はサーバーオーバーレイを使用することをお勧めします。サーバーオーバーレイは FFB デバイス上でサポートされます。サーバーオーバーレイについての詳細は、第 5 章を参照してください。

透明オーバーレイウィンドウとは

透明オーバーレイ拡張機能により、透明オーバーレイウィンドウを作成し、操作できます。このようなウィンドウは、ピクセル単位で操作することによりユーザーが背景のウィンドウを見ることができる X ウィンドウです。透明オーバーレイウィンドウを作成し、使用する機能はソフトウェアに実装されているので、特別なハードウェアは不要です。単純なハードウェア上で複雑な透明オーバーレイを操作するには時間がかかりますが、X サーバーが使用可能であれば特殊なオーバーレイハードウェアが使用できるので、クライアントは正しいビジュアルを選択します。ハー

ドウェアやニーズによっては、透明オーバーレイウィンドウに合わせてクライアントのカラー割り当てを調整しなければならないことがあるので注意してください。

オーバーレイウィンドウを使用すると、アプリケーションの表示ウィンドウに一時イメージを表示できます。オーバーレイを提供するアプリケーションのユーザーは、イメージにテキストや図表の注釈を付けたり、イメージの一部分を一時的に強調表示したり、イメージの上に動画を作成したりできます。オーバーレイ上のグラフィックスを消去しても、その下にあるグラフィックを生成し直す必要はありません。

透明オーバーレイ拡張機能により、クライアントは標準 X 要求を使用して不透明ペイントまたは透明ペイントでプリミティブを描画できます。不透明ペイントは標準的な描画方法の名称で、透明ペイントはピクセルを隠します。ペイント型は、標準 X グラフィックスコンテキストに関連付けられています。ウィンドウの背景も透明ペイントに設定できます。透明オーバーレイウィンドウは、すべての正規ウィンドウ規則と操作手順に準拠しています。たとえば、透明オーバーレイウィンドウは、それが関連付けられているハードウェアに関係なく、ウィンドウの重なり順序のどこにでも配置できます。これは、Solaris X サーバーの複数プレーングループ (MPG) 機能と一緒にソフトウェアに実装されています。

サーバーの複数プレーングループ機能により、異なるハードウェアから表示されるウィンドウを共存させることができます。各ウィンドウにはビジュアルが関連付けられており、ビジュアルはハードウェアに関連付けられています。一部のハードウェアは「層」の数が限定されるように物理的に作成されていますが (たとえば、ハードウェアのオーバーレイプレーン内で作成されたウィンドウは、常に正規ウィンドウの上に表示されるものと想定される)、MPG はソフトウェア内でこの制限を解決します。MPG により、ウィンドウの重なり順序はハードウェアの物理的制限の影響を受けなくなります。その結果、重なりは単に標準サーバー内と同じになります。オーバーレイハードウェアが使用可能であり、要求される場合は、MPG によって作業を最小限度に抑え性能を高めることができます。

一般に、オーバーレイはグラフィックスを描画できるピクセルバッファ (物理的またはソフトウェアによりシミュレート) です。オーバーレイが物理的なものである (つまり、ソフトウェアでシミュレートされたものではない) 場合は、オーバーレイ上のグラフィックスを消去しても、その下にあるグラフィックスは損傷を受けません。このため、下にあるグラフィックスが複雑で再描画に時間がかかる場合は、性能の向上に役立ちます。オーバーレイがソフトウェアに含まれる場合は、オーバーレイ上のグラフィックスを消去すると `Expose` イベントが生成されることがあります。

透明オーバーレイウィンドウの基本的な特性

透明オーバーレイウィンドウは、ピクセルを透明に描画できる X InputOutput ウィンドウの特殊なクラスです。透明オーバーレイウィンドウのハンドルは、X ウィンドウの型 Window を持っています。標準の X ウィンドウと同様に、オーバーレイウィンドウは描画可能であり、オーバーレイウィンドウのハンドルは Drawable を取る任意の Xlib 描画ルーチンに渡すことができます。

透明オーバーレイウィンドウによって、ペイント型の属性を含むグラフィックコンテキスト属性のセットが拡張されました。透明オーバーレイ拡張機能を使用すると、不透明ペイントまたは透明ペイントを使用して透明オーバーレイウィンドウを描画できます。

ペイント型

標準 X InputOutput ウィンドウや他のドロアブル (ピックスマップなど) には不透明ペイントしか使用できませんが、透明オーバーレイウィンドウでは透明ペイントを使用してピクセルを描画できます。ペイントされた有効なピクセル値により、背後のウィンドウ内のピクセルは不透明に隠されます。このようなピクセルには、表示されるカラー値が関連付けられています。透明に描画されたピクセルは固有のカラーを持たず、下にあるピクセルから表示カラーを継承します。

不透明にペイントされたピクセルに有効なピクセル値は、XAllocColor() または別の標準ピクセル割当メカニズムを介して取得されます。ビジュアルに有効なカラーマップエントリの範囲外の値など、有効でないピクセル値を使用して不透明にペイントすると、透明オーバーレイウィンドウの場合も標準 X InputOutput ウィンドウの場合も結果は不定になります。

ペイント型は、データ構造体 XSolarisOvlPaintType で定義されます。デフォルトでは、GC のペイント型は不透明です。XSolarisOvlPaintType データ構造体は次のように定義されます。

```
typedef

enum {  XSolarisOvlPaintTransparent,  XSolarisOvlPaintOpaque, }

XSolarisOvlPaintType;
```

可視性

透明オーバーレイウィンドウは、すべてのピクセルが透明であっても可視的であるとみなされます。透明オーバーレイウィンドウ内の完全に透明な可視ピクセルについては、その下にあるアンダーレイのピクセルが表示されます。

オーバーレイウィンドウがアンマップされたり移動されたりすると、その下のアンダーレイがエクスポージャーイベントを受け取る場合があります。たとえば、このような状況は、異なるプレーングループ内にあるオーバーレイウィンドウとアンダーレイウィンドウを同時に表示することができないデバイス上で発生します。

透明オーバーレイウィンドウのその他の特性

ほとんどの場合、透明オーバーレイウィンドウは標準 X `InputOutput` ウィンドウと似ています。特に、透明オーバーレイウィンドウには次のような特性があります。

- オーバーレイウィンドウに対してもマップとアンマップができます。そのために、`XMapWindow`、`XUnmapWindow`、`XMapSubwindows`、`XUnmapSubwindows` の各ルーチンが利用できます。
- オーバーレイウィンドウは独自のカーソルを所有するか、親ウィンドウのカーソルを使用します。すなわち、`XDefineCursor` と `XUndefineCursor` は、オーバーレイウィンドウにも適用されます。
- オーバーレイウィンドウは、`XQueryTree` の出力に表示されます。
- ウィンドウ属性の `event_mask` と `do_not_propogate_mask` は、正常に動作します。オーバーレイウィンドウは、任意の型のイベントを処理の対象とすることができます。
- `XTranslateCoordinates` と `XQueryPointer` は、オーバーレイウィンドウに適用されます。
- 標準の X ウィンドウについては、`save_under` が適用されます。
- 標準の X ウィンドウについては、`override_redirect` が適用されます。

また、透明オーバーレイウィンドウには、ウィンドウを固有のものにする特性があります。以下の節ではこのような特性について説明します。

背景

X 仕様で定義されているように、ウィンドウには背景を設定できます。ウィンドウの背景の主な目的は、クライアントがウィンドウの露出領域を再描画するのに時間がかかる場合に、何か適当なものをその領域に表示することです。ウィンドウが Expose イベントを受信する度に背景が描画されます。背景が描画された後で、Expose イベントがクライアントに送信されます。クライアントが XClearArea 要求または XClearWindow 要求を発行したときにも背景が描画されます。

標準の X InputOutput ウィンドウと同様に、透明オーバーレイウィンドウにも背景を設定できます。通常のウィンドウと同様に、透明オーバーレイウィンドウの背景は、Expose イベント、XClearArea 要求、XClearWindow 要求に応答して描画されます。標準タイプの背景 (None、pixmap、pixel、ParentRelative) の他に、透明オーバーレイウィンドウでは transparent という新しいタイプの背景を設定することもできます。透明 (transparent) の背景を指定する場合は、XSolarisOvlSetWindowTransparent という新しいルーチンを使用します。

デフォルトでは、透明オーバーレイウィンドウの背景は透明ですが、アプリケーション内では、表 6–1 に示されているように、通常の X タイプの背景 (None、ピックスマップ XID、ピクセル値、ParentRelative) を指定することもできます。

表 6–1 透明オーバーレイウィンドウの背景値

背景	説明
透明	透明オーバーレイウィンドウの背景はデフォルトで透明に描画される。
なし	オーバーレイウィンドウ内で背景の描画を起動する条件が検出されると、描画は実行されない。透明ペイントも不透明ペイントも描画されない。
ピックスマップ ID	背景は不透明ペイントで描画される。描画されるピクセル値は X 仕様で定義されたピックスマップから継承される。
単一のピクセル値	背景は不透明ペイントで一様なペイントのカラーで描画される。
ParentRelative	ParentRelative の背景の動作は、親ウィンドウの背景とタイプによって異なる。親ウィンドウがアンダーレイである場合、オーバーレイウィンドウである子の背景は不透明ペイントで描画され、ピクセルは X 仕様の定義に従って描画される。親ウィンドウがオーバーレイの場合、オーバーレイウィンドウの子の背景は、その親ウィンドウの背景と同じになり、透明ペイントまたは不透明ペイントで描画される。

XSolarisOvlSetTransparent を使用してオーバーレイ以外のウィンドウの背景を設定しようとすると、BadMatch エラーが発生します。アンダーレイウィンドウの背景が ParentRelative であり、親ウィンドウが透明な背景を持つオーバーレイである場合、アンダーレイウィンドウである子は None の背景を持つものとして取り扱われます。

ウィンドウの境界

オーバーレイウィンドウの境界は不透明です。境界は常に不透明なペイントで描画されます。標準の X InputOutput ウィンドウと同様に、境界の幅は XSetWindowBorderWidth を使用して制御することができます。

バッキングストア

バッキングストアはオーバーレイウィンドウについては無効です。

ウィンドウのグラビティ

ビットとウィンドウのグラビティ属性 (bit_gravity と win_gravity) は透明オーバーレイウィンドウにも適用されますが、グラビティによってピクセルの移動が必要になる場合は、ピクセルのカラー情報とともに透明かどうかという情報も移動されます。

カラーマップ

オーバーレイのカラーマップは、X の規則に従ってインストールされます。ピクセルを共有するオーバーレイ/アンダーレイのペアをアプリケーション内で使用する場合は、各ウィンドウごとに 1 つのカラーマップを作成してください。ピクセルを共有するウィンドウの問題については、75ページの「オーバーレイ/アンダーレイに使用するビジュアルの選択」と93ページの「アプリケーションの移植性の設計」を参照してください。

オーバーレイ/アンダーレイのペアがハードウェアカラー LUT を共有しないことがわかっている場合は、それらの両ウィンドウに別々のカラーマップを安全に割り当てることができ、カラーマップフラッシングは発生しません。

注 - アプリケーションの移植性を高め、カラーのフラッシングを最小限に抑えるために、オーバーレイとアンダーレイのカラーマップ内で同じカラーを使用してください。それができない場合は、フラッシングを発生させることなく異なるカラーマップを割り当てられるかどうかを、ビジュアル問い合わせルーチンを使用して調べてください。

入力配送モデル

オーバーレイウィンドウは、標準の X ウィンドウと同様にイベントを処理の対象とすることができます。オーバーレイウィンドウは、その実際に目に見える範囲内で発生する任意のイベントを受信します。イベントが発生する位置のピクセルのペイント型は問題ではありません。たとえば、オーバーレイウィンドウがウィンドウ進入イベントを処理の対象としている場合に、ポインタがウィンドウの実際に目に見える範囲内に入ると、ピクセルが透明であっても不透明であっても、オーバーレイウィンドウはウィンドウ進入イベントを受信します。

このことは、アプリケーション内でグラフィカルオブジェクトの対話的なピッキング (選択) をどのように実装するかという問題に影響を与えます。すなわち、アンダーレイウィンドウに描画されたグラフィカルオブジェクトの上にあるオーバーレイウィンドウに対して、別のグラフィカルオブジェクトを描画するアプリケーションは、オーバーレイウィンドウ内またはアンダーレイウィンドウ内のどちらかのイベントを処理の対象とする必要がありますが、その両方のウィンドウ内のイベントを処理の対象とする必要はありません。入力イベントを受け取ったアプリケーションは、オーバーレイ/アンダーレイの階層構造に関する知識を利用して、どちらのグラフィカルオブジェクトが選択されたかを確定しなければなりません。

たとえば、アプリケーションがアンダーレイウィンドウ内のイベントを処理の対象としていると想定します。このアプリケーションが座標 (x,y) でイベントを受信すると、まず最初にオーバーレイ内のその座標にグラフィカルオブジェクトが存在するかどうかが調べられます。存在すれば、そこで探索処理は終了します。存在しなければ、アプリケーションは、アンダーレイ内のその座標にグラフィカルオブジェクトが存在するかどうかを確認しなければなりません。

印刷用イメージの取り出し

グラフィカルイメージがX ウィンドウに描画されると、ユーザーはウィンドウの表示内容を取り出してプリンタに送信し、ハードコピーを取ることができます。そのための最も一般的な方法は、画面のダンプを実行することです。すなわち、`XGetImage` を使用してウィンドウピクセルを読み取り、そのイメージをプリンタに送信します。印刷ページのサイズにイメージを合わせるために、イメージの拡大・縮小が必要になる場合もあります。このため、イメージ内にピクセル単位の拡大・縮小によるがたつきが生じることがあります。

X11 の利用者の間で一般的になりつつあるもう一つの方法は、特殊なプリンタグラフィックス API を通じてグラフィックスを再描画することです。この API は標準の Xlib グラフィックスコールをサポートし、これらの関数呼び出しをページ記述言語 (PDL) 形式に変換して適切なプリントスプーラに送信します。この方法の利点は、走査変換が適用された後のピクセルではなく座標自身を拡大・縮小することによって、グラフィックスを印刷ページに合わせられることです。したがって、イメージのがたつきが最小限に抑えられます。

ただし、プリンタ API を使用する方法は、オーバーレイ/アンダーレイウィンドウのペアに適用するときには重大な障害があります。大部分の PDL では、不透明なペイントの概念だけがサポートされ、透明なペイントはサポートされません。たとえば PostScript PDL の場合、以前に出力されたピクセルは新たに出力されたピクセルによって常に置き換えられます。このような制限があるため、プリンタ API を使用する方法では、オーバーレイ/アンダーレイウィンドウの組み合わせ中のイメージを完全に取り出せるとは限りません。特に、オーバーレイの背景が完全に透明で、不透明なペイントだけがそれに描画されるような特殊なアプリケーションでは、アンダーレイが最初に出力された後、オーバーレイが 出力されます。ただし、透明なペイントがオーバーレイに描画されて、そのオーバーレイ内の他の不透明なペイントが消去されると、上記の仕組みはうまく動作しません。

このような問題が解決されるまでは、`XReadScreen` と拡大・縮小処理を使用して、オーバーレイウィンドウ内のイメージの取り出しとプリンタへの送信を行ってください。あるいは、プリントする予定の情報を描画するときは、オーバーレイを使用しないでください。

オーバーレイ/アンダーレイに使用するビジュアルの選択

Solaris 透明オーバーレイ API では、複数プレーングループ (MPG) デバイスと単一プレーングループ (SPG) デバイスがサポートされます。表示デバイスにはさまざまな構成があります。表示デバイスによっては、複数プレーングループを持つものがあります。複数のハードウェアカラールックアップテーブル (LUT) を持つものもあります。また、特定のプレーングループ専用にカラー LUT を使用するものもあれば、複数のプレーングループ間でカラー LUT を共有するものもあります。このような広範な構成のために、アプリケーション開発者が移植性のあるオーバーレイアプリケーションを作成するのが困難になっています。

特定のタイプのアンダーレイウィンドウについては、高性能の描画能力を持つオーバーレイウィンドウが、特定のデバイスによって提供される場合がありますが、それより描画速度の遅い従来のオーバーレイウィンドウを提供するデバイスもあります。デバイスの中には、数多くのカラーを持つオーバーレイをサポートできるものもあればサポートできないものもあります。また、すべてのタイプのオーバーレイとアンダーレイのカラーを同時に表示できるデバイス、特定のオーバーレイ/アンダーレイの組み合わせについてはカラーの同時表示ができないデバイス、カラーの同時表示に若干の制限があるデバイスなどがあります。これらのデバイスは、複数のハードウェアカラー LUT をサポートします。ただし、ハードウェア内のカラー LUT の数が足りないために、アプリケーションによってはカラーを同時に表示できない場合があります。

Solaris Visual Overlay Window API では、アプリケーションがオーバーレイ/アンダーレイの最適なビジュアルの組み合わせについてシステムと折衝できるように、次の 2 つのユーティリティルーチンが提供されています。

- `XSolarisOvlSelectPartner`
- `XSolarisOvlSelectPair`

これらのルーチンについての詳細は、93ページの「アプリケーションの移植性の設計」を参照してください。

システムとの折衝では、各アプリケーションについて、ウィンドウとカラーの理想的な構成があることが前提とされます。アプリケーションはまず最初に、オーバーレイ/アンダーレイの「最適な」組み合わせを要求します。最初は理想的なペアを要求するという点で、アプリケーションは「最適」の定義について非常に大胆です。この要求がデバイスによって満足されれば、折衝は成立し、アプリケーションは選

択されたオーバーレイ/アンダーレイのビジュアルに基づいてウィンドウを作成します。要求を満たすビジュアルの組み合わせが存在しない場合、アプリケーションは要求を緩和し、「2 番目に最適な」組み合わせを指定します。たとえば、アプリケーションは表示可能な色数の少ないビジュアルを選択したり、特定のビジュアルについては要求する描画性能水準を落としたりします。このような手続きは、満足のゆくビジュアルが見つかるか、特定の条件が満たされない現在の環境で実行するのは意味がないとアプリケーションが判断するまで続けられます。

透明オーバーレイ API には、アプリケーションがこのような折衝を単一のサブルーチン呼び出しで実行できるようにするルーチンがあります。アプリケーションは、オーバーレイビジュアルとアンダーレイビジュアルの一方またはそれら両方について満足する条件を指定します。広範囲のグラフィックス装置に対する移植性を保障するために、アプリケーション内ではこれらのルーチンを使用することをお薦めします。

プログラム例

以下のプログラムは、透明オーバーレイの単純な例を示しています。このプログラムは透明オーバーレイウィンドウを作成し、ウィンドウのボーダを白で描画し、テキスト文字列を白で表示し、白で塗りつぶされた矩形を描画します。ペイント型はデフォルトで不透明に描画され、ウィンドウの背景はデフォルトで透明に描画されます。次の `Makefile` を使用して、このプログラムをコンパイルし、リンクしてください。

```
simple:

simple.c   cc -I../ -I/usr/openwin/include -o simple simple.c \

 -L/usr/openwin/lib -lX11 -lXext
```

例 6–1 Transparent Overlay Example Program

```
#include <stdio.h>

 #include <X11/Xlib.h>  #include ‘‘X11/Xmd.h’’  #include

<X11/extensions/transovl.h>  #include

<X11/extensions/transovlstr.h> Display       *display;  Window

      window;  XSetWindowAttributes       attribs; GC       gc;  XGCValues

      gcvalues; main()  {   display = XOpenDisplay(‘‘‘‘);
```

```
attribs.override_redirect = True;   attribs.border_pixel = WhitePixel(display,
0); window = XSolarisOvlCreateWindow(display,      DefaultRootWindow(display),
     100, 100, 500, 500, 10,      CopyFromParent, InputOutput,CopyFromParent,
    CWBorderPixel | CWOverrideRedirect, &attribs); gcvalues.font =
XLoadFont(display, ''fixed'');   gcvalues.foreground =
WhitePixel(display, 0); gc = XCreateGC(display, window, GCFont | GCForeground,
&gcvalues);   XMapWindow(display, window); XDrawString(display, window,
gc, 50, 50, ''This is a test'', 14);   XFillRectangle(display,
window, gc, 70, 70, 100, 100); XFlush(display); while   (1);}
```

Solaris 透明オーバーレイウィンドウ API の概要

透明オーバーレイウィンドウ API には、表 6–2 に示すルーチンが含まれています。これらのルーチンは `libXext.so` によって提供されます。Solaris オーバーレイルーチンを使用するには、次の操作を実行してください。

- `/usr/openwin/include/X11/extensions/transovl.h` ファイルをインクルードする。
- `/usr/openwin/lib/libXext.so` ライブラリにライブラリのデバイスハンドラをリンクする。

表 6–2 透明オーバーレイウィンドウルーチンのリスト

名前	説明
`XSolarisOvlCreateWindow`	オーバーレイウィンドウを作成する。
`XSolarisOvlIsOverlayWindow`	ウィンドウがオーバーレイウィンドウかどうかを示す。
`XSolarisOvlSetPaintType`	後続の Xlib 描画によって描画されるペイントの型を指定する。
`XSolarisOvlGetPaintType`	現在のペイント型を取得する。

表 6–2　透明オーバーレイウィンドウルーチンのリスト　続く

名前	説明
XSolarisOvlSetWindowTransparent	オーバーレイウィンドウの背景の状態を透明に設定する。
XSolarisOvlCopyPaintType	ソースドロアブル内のピクセルのペイント型属性に基づいて、宛先ドロアブルに不透明ペイントと透明ペイントを描画する。
XSolarisOvlCopyAreaAndPaintType	あるドロアブルのペアから別のペアに領域とペイント型をコピーする。
XReadScreen	画面の矩形に表示されるカラーを戻す。
XSolarisOvlSelectPartner	既存のビジュアルに最適のオーバーレイビジュアルまたはアンダーレイビジュアルを戻す。
XSolarisOvlSelectPair	オーバーレイビジュアルとアンダーレイビジュアルに定義された基準セットに最適のオーバーレイ / アンダーレイのペアを選択する。

この章の残りの節では、透明オーバーレイ API のルーチンについて説明します。

透明オーバーレイウィンドウの作成

XSolarisOvlCreateWindow を使用すると、透明オーバーレイウィンドウを作成できます。このルーチンは、XCreateWindow と同様に動作しますが、生成されるのは透明オーバーレイウィンドウです。新しく作成されたウィンドウを不透明ペイントと透明ペイントで描画でき、オーバーレイウィンドウの背景は透明になります。

XSolarisOvlCreateWindow の class 引数は、InputOutput にする必要があります。オーバーレイウィンドウは InputOnly ウィンドウとして作成できますが、その場合は標準 InputOnly ウィンドウと同様に動作します。オーバーレイと非オーバーレイに違いがあるのは、InputOutput ウィンドウだけです。

XSolarisOvlCreateWindow の構文と引数を次に示します。

```
Window
XSolarisOvlCreateWindow(Display *display, Window parent, int x, int y,
  unsigned int width, unsigned int height,   unsigned int border_width, int
depth, unsigned int class,   Visual * visual, unsigned long valuemask,
XSetWindowAttributes * attr)
```

このルーチンの引数は、XCreateWindow と同じです。

表 6–3

display	X サーバーへの接続を指定する。
parent	親ウィンドウを指定する。
x, y	ウィンドウの左上隅のピクセルの座標を親ウィンドウに対する相対座標で指定する。
width, height	ウィンドウの幅と高さをピクセル単位で指定する。
border_width	ウィンドウの境界の幅をピクセル単位で指定する。
depth	ウィンドウのデプスを指定する。
class	ウィンドウのクラスを指定する。クラスが InputOutput でない場合、ウィンドウはオーバーレイウィンドウにはならない。
visual	ウィンドウのビジュアル構造体へのポインタを指定する。
valuemask	attr 引数でどのウィンドウ属性を定義するかを指定する。
attr	ウィンドウ属性を指定する。

任意のビジュアルを使用してオーバーレイを作成できます。ただし、どのオーバーレイ / アンダーレイのペアでも最適であるとは限りません。画面ごとに、最適のオーバーレイ / アンダーレイビジュアルのペアのセットを定義します。これらは、特定のアンダーレイビジュアルを使用して作成できるオーバーレイウィンドウの最適のビジュアルを定義します。また、特定のオーバーレイビジュアルを使用して作成できるアンダーレイウィンドウの最適のビジュアルも定義しま

す。XSolarisOvlSelectPair と XSolarisOvlSelectPartner を使用すると、最適のペアを設定できます。

「最適」の定義はデバイスごとに異なりますが、通常はデバイスでアンダーレイウィンドウとは異なるプレーングループ内にオーバーレイウィンドウを作成できるかどうかを指します。オーバーレイ / アンダーレイビジュアルのペアについての詳細は、98ページの「オーバーレイ/アンダーレイビジュアルの最適なペアの選択」を参照してください。

オーバーレイウィンドウは、Xlib ルーチンである XDestroywindow または XDestroySubwindows によって削除されます。

グラフィックスコンテキストのペイント型の設定

XSolarisOvlSetPaintType ルーチンを使用して、GC のペイント型を設定できます。XSolarisOvlSetPaintType では、指定された GC を使用して後続の Xlib 描画によって描画されるペイントの型を指定します。このルーチンは、この GC を使用する Xlib 描画ルーチンによってオーバーレイウィンドウ上に不透明ピクセルが生成されるか、透明ピクセルが生成されるかを制御します。指定したペイント型は、このルーチンの別の呼び出しによって変更されるまで、GC に適用されます。ペイント型属性は、前景と背景の GC 属性に適用されます。構文と引数を次に示します。

```
void
XSolarisOvlSetPaintType (Display *display, GC gc,   XSolarisOvlPaintType
paintType)
```

display	X サーバーへの接続を指定する。
gc	対象の GC を指定する。
paintType	指定された GC を使用して、以降の Xlib 描画ルーチンで描画されるペイントの型を指定する。

paintType の値として、XSolarisOvlPaintOpaque、XSolarisOvlPaintTransparent のどちらかを指定できます。

- paintType の値が XSolarisOvlPaintOpaque の場合は、指定された GC を使用する Xlib 描画ルーチンで生成されるピクセルは不透明になります。したがっ

て、下にあるピクセルは上のピクセルによって隠されます。これは、デフォルトです。

- paintType の値が XSolarisOvlPaintTransparent である場合は、指定された GC を使用する Xlib 描画ルーチンで生成されるピクセルは透明になります。したがって、下にあるピクセルの色が表示されます。

透明オーバーレイウィンドウの背景の設定

XSolarisOvlSetWindowTransparent ルーチンを使用して、透明オーバーレイウィンドウの背景状態を透明に設定できます。この要求の後に描画される背景は、すべて透明になります。背景状態を他の値に変更するには、XChangeWindowAttributes()、XSetWindowBackground()、または XSetWindowBackgroundPixmap() のいずれかを使用します。

XSolarisOvlSetWindowTransparent の構文と引数を次に示します。

```
void

XSolarisOvlSetWindowTransparent (Display *display, Window

w)
```

display	X サーバーへの接続を指定する。
w	透明オーバーレイウィンドウを指定する。

注 - w が透明オーバーレイウィンドウでない場合は、BadMatch エラーが発生します。

透明オーバーレイウィンドウへの描画

透明オーバーレイウィンドウの作成が終わったら、XDrawLines や XFillRectangles など、すべての標準 Xlib プリミティブ描画ルーチンを使用してウィンドウに描画できます。透明オーバーレイウィンドウに描画するときには、GC のペイント型属性を使用して、描画されるピクセルの品質が制御されます。

ペイント型属性は、前景と背景の GC 属性に適用されます。ペイント型を設定するには、XSolarisOvlSetPaintType ルーチンを使用します。このルーチンについての詳細は、80ページの「グラフィックスコンテキストのペイント型の設定」を参照してください。

GC のペイント型は、XPutImage で描画されるピクセルの型も制御します。引数 GC のペイント型が XSolarisOvlPaintOpaque の場合は、ソースイメージからのカラー情報が使用され、ピクセルは不透明ペイントで描画されます。ただし、ペイント型が XSolarisOvlPaintTransparent の場合は、ソースカラー情報が無視され、ピクセルは透明ペイントで描画されます。

ペイント型が XSolarisOvlPaintTransparent である GC を使用して、アンダーレイウィンドウやピックスマップなど、透明オーバーレイウィンドウ以外のドロアブルに描画する場合は、GC ペイント型が無視され、ピクセルは不透明ペイントで描画されます。

透明オーバーレイウィンドウの特性の問い合わせ

XSolarisOvlIsOverlayWindow ルーチンを使用して、ウィンドウがオーバーレイウィンドウかどうかを判断できます。また、XSolarisOvlGetPaintType ルーチンを使用して、GC の現在のペイント型を判断できます。

ウィンドウがオーバーレイウィンドウかどうかの判断

XSolarisOvlIsOverlayWindow ルーチンを使用して、ウィンドウがオーバーレイウィンドウかどうかを判断できます。このルーチンは、指定されたウィンドウ w がオーバーレイウィンドウの場合は True を戻し、それ以外の場合は False を戻します。

```
Bool

XSolarisOvlIsOverlayWindow (Display *display, Window

w)
```

display	X サーバーへの接続を指定する。
w	ウィンドウを指定する。

グラフィックスコンテキストのペイント型の判断

XSolarisOvlGetPaintType ルーチンは、GC の現在のペイント型を戻します。

```
XSolarisOvlPaintType
XSolarisOvlGetPaintType (Display *display, GC
gc)
```

display	X サーバーへの接続を指定する。
gc	問い合わせ対象の GC を指定する。

ピクセル転送ルーチン

透明オーバーレイ API は、次の 3 つのピクセル転送ルーチンを提供します。

- XSolarisOvlCopyPaintType – ソースドロアブルのペイント型属性に基づいて、宛先ドロアブルに不透明ポイントと透明ポイントを描画する。
- XSolarisCopyAreaAndPaintType – ドロアブルの一方のペアから別のペアに領域とそのペイント型をコピーする。
- XReadScreen – 指定された画面領域に表示されるカラーを戻す。

既存の Xlib ピクセル転送ルーチン XGetImage、XCopyArea、XCopyPlane も、オーバーレイウィンドウに使用できます。これらのルーチンの使用方法については、以下の節で説明します。

ソースエリアペイント型を使って描画する

XSolarisOvlCopyPaintType ルーチンは、ソース矩形内の指定された矩形のペイント型情報を使用して、宛先矩形内の指定された矩形の塗りつぶし操作を制御します。ソース矩形と宛先矩形には、任意のタイプのドロアブルを指定できます。ソース矩形が透明オーバーレイの場合は、そのピクセルのペイント型属性がコピーソースとして使用され、カラー情報は無視されます。ソース矩形が他のタイプのドロアブルであれば、ルーチン内で指定されたビットプレーンはペイント型データであるかのように処理され、コピーに使用されます。この場合、ビットプレーンはビットセットを 1 つだけ持たなければなりません。

構文と引数を次に示します。

```
void
XSolarisOvlCopyPaintType(Display *display, Drawable src,
  Drawable dst, GC gc, int src_x, int src_y,
  unsigned int width, unsigned int height, int dest_x,
  int dest_y, unsigned long action, unsigned long
plane)
```

display	X サーバーへの接続を指定する。
src	ペイント型属性に関する情報を取り出すソースのドロアブル (Drawable) を指定する。
dst	宛先のドロアブルを指定する。
gc	GCを指定する。
src_x, src_y	ソース矩形の左上隅の xy 座標をソースドロアブルの原点に対する相対座標で指定する。
width, height	ソースおよび宛先の矩形の幅と高さを指定する。
dest_x, dest_y	宛先矩形の左上隅の xy 座標を宛先ドロアブルの原点に対する相対座標で指定する。

action	コピーするペイント型データを指定する。指定できる値は、XSolarisOvlCopyOpaque、XSolarisOvlCopyTransparent、または XSolarisOvlCopyAll のいずれか。
plane	ソースが透明オーバーレイでないときにペイント型情報として使用する src ドロアブルのビットプレーンを指定する。

src と dst は同じスクリーンを持たなければなりません。そうしないと、BadMatch エラーが発生します。

表 6–4に、src と dst の組み合わせとその動作を示します。表の左側は src の組み合わせ、表の上側は dst の組み合わせを表します。A1～A4 の各動作については、表に続いて説明します。

表 6–4 XSolarisOvlCopyPaintType のソース/宛先の組み合わせと動作

ソース/宛先	オーバーレイ	ドロアブル
オーバーレイ	A1	A2
ドロアブル	A3	A4

- A1 — ソースオーバーレイの不透明ピクセルに対応する宛先ピクセルは、GC の塗りつぶし属性で指定された不透明カラーで描画されます。ソースオーバーレイの透明ピクセルに対応する宛先ピクセルは、透明なペイントで描画されます。
- A2 — ソースオーバーレイの不透明ピクセルに対応する宛先ピクセルは、GC の塗りつぶし属性に従って描画されます。ソースオーバーレイの透明ピクセルに対応する宛先ピクセルは、GC の同じ塗りつぶし属性に従って描画されますが、前景と背景のピクセルが交換されます。
- A3 — 宛先オーバーレイのピクセルは、ソースドロアブルの plane のビット値に応じて (上記の A1 と同様に) 不透明ペイントまたは透明ペイントで描画されます。ソースの 1 というビット値は不透明ピクセルとして取り扱われ、0 というビット値は透明ピクセルとして取り扱われます。
- A4 — 宛先ドロアブルのピクセルは、ソースドロアブルの plane のビット値に応じて (上記の A2 と同様に) 描画されます。ソースのビットプレーンの 1 というビット値は不透明ピクセルとして取り扱われ、0 というビット値は透明ピクセルとして取り扱われます。

action 引数は、不透明ペイント (XSolarisOvlCopyOpaque) 、透明ペイント (XSolarisOvlCopyTransparent) 、それら両方 (XSolarisOvlCopyAll) のいずれをコピー対象とするかを指定します。これによって、クライアントは不透明ペイントまたは透明ペイントを累積することができます。

ソース矩形の一部が、隠れたりソースドロアブルの境界外部にある場合、サーバーは XCopyArea と同じセマンティクスを使用して Expose イベントを生成します。

このルーチンで使用される GC 構成要素は、function、plane-mask、fill-style、subwindow-mode、graphics-exposures、clip-x-origin、clip-y-origin、clip-mask です。また、GC モードに依存する構成要素として、foreground、background、tile、stipple、tile-stipple-x-origin、tile-stipple-y-origin が使用される場合もあります。

このルーチンで発生する可能性のあるエラーは、BadDrawable、BadGC、BadMatch、BadValue です。

エリアとそのペイント型のコピー

XSolarisCopyAreaAndPaintType ルーチンは、カラー情報のソースとなるドロアブルの指定された領域を、カラー情報の宛先イメージとなるドロアブルの指定された領域にコピーします。宛先イメージとなるドロアブルがオーバーレイでなければ、ペイント型情報のソースとなるドロアブル内で指定されたペイント型情報に従って、ペイント型情報の宛先イメージとなるドロアブルの指定された領域も塗りつぶします。

XSolarisOvlCopyAreaAndPaintType ルーチンを使用すれば、カラーまたはペイント型の情報が格納されているクライアントのメモリー空間内のイメージと、指定されたオーバーレイウィンドウの矩形とを結合することができます。その場合は、最初にイメージとペイント型のデータをサーバーに移動し、XPutImage を使用して、適切なデプスの 2 つのピクセルマップにデータをコピーします。次に、カラーとペイント型のドロアブルを指定した XSolarisOvlCopyAreaAndPaintType を呼び出して、情報をオーバーレイにコピーします。

このルーチンを使用すれば、特定のドロアブルからピクセル情報 (カラーとペイント型の情報) を取り出すこともできます。その場合は、最初に 2 つの分離可能な宛先ドロアブルを指定する XSolarisOvlCopyAreaAndPaintType を呼び出します。次に、各ドロアブルに対して XGetImage を呼び出し、サーバーからクライアントのメモリー空間にデータを転送します。

XSolarisCopyAreaAndPaintType の構文と引数を次に示します。

```
void
XSolarisOvlCopyAreaAndPaintType(Display * display, Drawable colorsrc,
  Drawable painttypesrc, Drawable colordst,   Drawable painttypedst, GC
colorgc, GC painttypegc,   int colorsrc_x, int colorsrc_y, int
painttypesrc_x,   int painttypesrc_y, unsigned int width,
unsigned int height,   int colordst_x, int colordst_y, int
painttypedst_x,   int painttypedst_y, unsigned long action, unsigned long
plane)
```

display	X サーバーへの接続を指定する。
colorsrc	カラー情報のソースとなるドロアブル。colorsrc として、任意のデプスのドロアブルまたはオーバー レイウィンドウを指定できる。
painttypesrc	ペイント型情報のソースとなるドロアブル。painttypesrc として、任意のドロアブルまたはオーバーレイウィンドウを指定できる。painttypesrc がオーバーレイウィンドウでない場合、plane で指定された painttypesrc のビットプレーンはペイント型データであるかのように処理され、コピーに使用される。この場合、plane は 1 つのビットセットを持たなければならない。
colordst	カラー情報の宛先となるドロアブル。
painttypedst	ペイント型情報の宛先となるドロアブル。colordst がオーバーレイである場合、このドロアブルは無視される。
colorgc	カラー情報のコピーに使用する GC。
painttypegc	painttypedst 内の領域の描画に使用する GC。colordst/painttypedst がオーバーレイである場合、この GC は無視される。
colorsrc_x colorsrc_y	カラー情報のソース矩形の左上隅の XY 座標を、カラー情報のソースドロアブルの原点に対する相対座標で指定する。
painttypesrc_x painttypesrc_y	ペイント型情報のソース矩形の左上隅の XY 座標を、ペイント型情報のソースドロアブルの原点に対する相対座標で指定する。

width, height	ソースおよび宛先の矩形の幅と高さをピクセル単位で指定する。
colordst_x colordst_y	カラー情報の宛先矩形の左上隅の XY 座標を、カラー情報の宛先ドロアブルの原点に対する相対座標で指定する。
painttypedst_x painttypedst_y	ペイント型情報の宛先矩形の左上隅の XY 座標を、ペイント型情報の宛先ドロアブルの原点に対する相対座標で指定する。colordst/painttypedst がオーバーレイである場合は、colordst_x と colordst_y が使用される。
action	どのペイント型データをコピーするかを指定する。指定できる値は、XSolarisOvlCopyOpaque、XSolarisOvlCopyTransparent、XSolarisOvlCopyAll のいずれか。
plane	painttypesrc がオーバーレイでないときに、ペイント型情報として使用する painttypesrc のソースビットプレーンを指定する。

colordst として任意のドロアブルを指定できますが、colorsrc と同じデプスで同じルートを持たないと、BadMatch エラーが発生します。colordst がオーバーレイの場合は、painttypedst が無視され、それ以外の場合は painttypedst として任意のタイプのドロアブルを指定できます。

表 6–5 に、ソースと宛先の組み合わせとそれらの動作を示します。表の左側は colorsrc/paittypesrc の組み合わせ、表の上側は colordst/painttypedst の組み合わせを表します。A1～A8 の各動作については、表に続いて説明します。表の中で「不可能」と記載された箇所は、colordst がオーバーレイの場合は painttypedst が無視されるために、その組み合わせが不可能であることを示します。

表 6–5 XSolarisOvlCopyAreaAndPaintType のソース/宛先の組み合わせと動作

	オーバーレイ/オーバーレイ	オーバーレイ/ドロアブル	ドロアブル/オーバーレイ	ドロアブル/ドロアブル
オーバーレイ/オーバーレイ	A1	不可能	A5	A5
オーバーレイ/ドロアブル	A2	不可能	A6	A6

表 6–5　XSolarisOvlCopyAreaAndPaintType のソース/宛先の組み合わせと動作　続く

	オーバーレイ/オーバーレイ	オーバーレイ/ドロアブル	ドロアブル/オーバーレイ	ドロアブル/ドロアブル
ドロアブル/オーバーレイ	A3	不可能	A7	A7
ドロアブル/ドロアブル	A4	不可能	A8	A8

- A1 — painttypesrc のペイント型情報が、colorsrc から colordst にカラー情報をコピーするためのマスクとして使用されます。painttypesrc 内の不透明ピクセルに対応する colorsrc のピクセルは colordst にコピーされ、透明ピクセルに対応する colordst のピクセルは透明になります。colorsrc の透明ピクセルが colordst にコピーされる場合、実際に転送されるカラーは未定義になります。
- A2 — plane で指定される painttypesrc のビットプレーンからペイント型情報が抽出される点以外は、A1 と同じです。1 というビット値は不透明ピクセルを表し、0 というビット値は透明ピクセルを表します。
- A3 — オーバーレイでないドロアブルを使用してカラー情報が取得される点以外は、A1 と同じです。透明ピクセルを原因とする未定義のカラーは生じません。
- A4 — A2 と同様に、 plane で指定される painttypesrc のビットプレーンからペイント型情報が抽出される点以外は、A3 と同じです。
- A5 — A1 と同様に、 painttypesrc のペイント型情報が、colorsrc から colordst にカラー情報をコピーするためのマスクとして使用されます。さらに、XSolarisOvlCopyPaintType の場合のように、このペイント型情報によって painttypedst ドロアブルに対する描画が制御されます。
- A6 — A2 と同様に、plane で指定される painttypesrc のビットプレーンからペイント型情報が抽出される点以外は、A5 と同じです。
- A7 — カラー情報のソースの透明ピクセルを原因とする未定義のカラーが生じない点以外は、A5 と同じです。
- A8 — A2 と同様に、 plane で指定される painttypesrc のビットプレーンからペイント型情報が抽出される点以外は、A7 と同じです。

action 引数は、不透明ペイント (XSolarisOvlCopyOpaque) 、透明ペイント (XSolarisOvlCopyTransparent) 、それら両方 (XSolarisOvlCopyAll) のいず

れをコピー対象とするかを指定します。これによって、クライアントは不透明ペイントまたは透明ペイントを累積することができます。

XSolarisOvlCopyPaintType と同様の方法で、NoExpose イベントと GraphicsExpose イベントが生成されます。

colordst 引数にオーバーレイを指定すると、painttypedst、painttypegc、painttypedst_x、painttypedst_y の各引数はすべて無視されます。painttypegc には NULL ポインタ、painttypedst には None 値を指定できます。オーバーレイは、painttypesrc で指定される領域内のピクセルで定義されるのとまったく同じペイント型を持ちます。カラー情報をコピーしても、宛先のペイント型は影響を受けません。

このルーチンで使用される colorgc の GC 構成要素は、function、plane-mask、subwindow-mode、graphics-exposure、clip-x-origin、clip-y-origin、clip-mask です。

colordst がオーバーレイでない場合は、painttypegc の GC 構成要素として、function、plane-mask、fill-style、subwindow-mode、clip-x-origin、clip-y-origin、clip-mask が使用されます。また、GC モードに依存する構成要素として、foreground、background、tile、stipple、tile-stipple-x-origin、tile-stipple-y-origin が使用される場合もあります。

このルーチンで発生する可能性のあるエラーは、BadDrawable、BadGC、BadMatch、BadValue です。

オーバーレイカラー情報の検索

XReadScreen は、スクリーンの矩形に表示されるカラーを戻します。このルーチンは、指定されたウィンドウの画面に表示されるカラーをアクセスします。

一部の高度な表示デバイス上では、表示されるカラーがデータの複合体として、複数の異なるフレーム記憶域に格納されており、これらのフレーム記憶域が異なるデプスやビジュアル型を持つことができます。また、アンダーレイの一部分がオーバーレイの下に見えるようなオーバーレイ/アンダーレイのウィンドウペアもあります。デプスが異なる矩形については、 XGetImage によって戻されるデータが未定義になるため、XGetImage は、ユーザーが実際に画面上で見ている画像を返すには不十分です。さらに、ピクセル情報が異なるドロアブルに存在するため、XGetImage はオーバーレイ/アンダーレイのウィンドウペアに関するピクセル情報を合成することができません。XReadScreen がこのような問題を解決します。

XReadScreen は、ピクセル情報ではなくカラー情報 (画面上に実際に表示される可視カラー) を戻します。すなわち、指定された矩形の境界内部のすべてのウィンドウに関するカラー情報を戻します。XGetImage とは違って、指定されたウィンドウのデプスとは異なるデプスを持つ下層ウィンドウやオーバラップウィンドウであっても、その可視領域について戻される情報は未定義にはならず、これらのウィンドウで実際に表示されるカラーが戻されます。

注 - 戻されるカラーは、画面上で利用できるハードウェアカラー LUT の数に制限がないと想定した場合には、表示されるカラーとなります。したがって、このカラーは理論的な表示カラーを意味します。すべてのソフトウェアカラーマップを同時に表示するのに十分な数のハードウェアカラー LUT がないと、画面上でカラーマップフラッシングが発生しますが、この場合は、戻されるカラーと実際に表示されるカラーが一致しないことがあります。

このルーチンの構文と引数を次に示します。

```
XImage

* XReadScreen (Display *display, Window w, int x, int y,
   unsigned int width, unsigned int height,

  Bool includeCursor)
```

display	X サーバーへの接続を指定する。
w	スクリーンのデータが読み取られるウィンドウを指定する。
x, y	矩形の左上隅の XY 座標をウィンドウ w の原点に対する相対座標で指定する。
width, height	矩形の幅と高さを指定する。
includeCursor	戻されるカラーにカーソルイメージを含めるかどうかを指定する。

w がオーバーレイウィンドウである場合、指定された矩形内に不透明なペイントが存在するすべての領域については、オーバーレイのカラー情報が戻されます。オーバーレイ内に透明なペイントが存在する領域については、アンダーレイのカラー情報が戻されます。通常、このアンダーレイは透明ペイントを含むオーバーレイウィンドウにできるため、透明ペイントを含む (x, y) 座標に関するカラー情報は、(x, y) に不透明なペイントを持つ最初に出合った上層ウインドウを表します。

カラー情報は、XImage として戻されます。戻されるイメージの幅と高さは、引数で指定された値と同じになります。イメージの形式は ZPixmap です。イメージのデプスは 24、bits_per_pixel は 32 です。各カラーチャネル (赤、緑、青) に関するカラー情報の最上位 8 ビットは、XImage 内の red_mask、green_mask、blue_mask で定義されるビット位置に戻されます。XImage の属性値のうちサーバーに依存するものは、byte_order, bitmap_unit, bitmap_bit_order, bitmap_pad, bytes_per_line, red_mask, green_mask, blue_mask です。

includeCursor が True の場合は、戻されるカラーにカーソルイメージが含められ、それ以外の場合は、除外されます。

このルーチンでは、引数のウィンドウ (およびその他のウィンドウ) の境界も読み取られることに注意してください。

問題が発生すると、XReadScreen は NULL を戻します。

既存の Xlib ピクセル転送ルーチンを使用する

Xlib ピクセル転送ルーチン XGetImage、XCopyArea、XCopyPlane も、透明オーバーレイウィンドウに使用できます。

XGetImage

オーバーレイ以外のドロアブル上では、XGetImage ルーチンは X11 仕様で定義されたとおりに動作します。オーバーレイウィンドウについても同じことが言えますが、例外として、これらのウィンドウ上では透明なピクセルについて戻されるカラー情報は未定義になります。画面上の領域の表示カラーを検索するだけの場合は、XReadScreen を使用してください。

XCopyArea および XCopyPlane

ソースと宛先のドロアブルが両方ともオーバーレイ以外の場合、これらのルーチンは X11 仕様で定義されたとおりに動作します。ただし、ソースまたは宛先となるドロアブルがオーバーレイウィンドウの場合は、次の点に注意してください。

- ソースのドロアブルがオーバーレイで、宛先のドロアブルがオーバーレイ以外の場合は、カラー情報だけがコピーされ、ソースのペイント型情報は無視されます。透明なピクセルに関するカラー情報は未定義になります。

- ソースのドロアブルがオーバーレイ以外で、宛先のドロアブルがオーバーレイである場合は、GC のペイント型で指定されたとおりにコピーが実行されます。すなわち、ペイント型が XSolarisOvlPaintOpaque の場合は、不透明なペイントを使用してカラー情報が宛先にコピーされます。ペイント型が XSolarisOvlPaintTransparent の場合は、カラー情報が無視され、宛先のピクセルは透明になります。
- ソースと宛先のドロアブルが両方ともオーバーレイの場合は、ソースのペイント型が無視され、ソースがオーバーレイでない場合と同じ動作になります。カラーとペイント型の両方の情報をコピーしたい場合は、XSolarisOvlCopyAreaAndPaintType を使用してください。

アプリケーションの移植性の設計

Solaris オーバーレイ API には、デバイス間でアプリケーションの移植性を保証できるように、次の 2 つのルーチンが用意されています。

- XSolarisOvlSelectPartner – アプリケーションで既存のオーバーレイビジュアルまたはアンダーレイビジュアルに最適の組み合わせになるビジュアルを選択できるようにする。
- XSolarisOvlSelectPair – アプリケーションで、画面用のすべてのビジュアルペアセットから最適のオーバーレイビジュアルとアンダーレイビジュアルのペアを選択できるようにする。

次に、この 2 つのルーチンについて説明します。

オーバーレイ/アンダーレイウィンドウのビジュアルの選択

オーバーレイを使用するアプリケーションの移植性を維持するためには、適切なオーバーレイビジュアルを探索し、指定されたアンダーレイビジュアルに使用できることが必要です。あるいは適切なアンダーレイビジュアルを探索し、指定されたオーバーレイビジュアルに使用できることが必要です。オーバーレイ拡張機能をサポートする各 X スクリーンでは、アンダーレイウィンドウの子として使用するのに最適なウィンドウを持つオーバーレイビジュアルの集合が定義されます。各アンダーレイビジュアルごとに最適なオーバーレイビジュアルの集合が存在します。アンダーレイビジュアルとそれらに最適なオーバーレイビジュアルの組み合わせ

によって、各スクリーンの「オーバーレイ/アンダーレイの最適なペア」が形成されます。最適なペアを形成するオーバーレイ/アンダーレイの各ビジュアルは、お互いの「パートナ」と呼ばれます。

ルーチン XSolarisOvlSelectPartner を使用すれば、アンダーレイビジュアルを入力として与えて、特定の条件を満たす最適なオーバーレイビジュアルを選択することができます。またその逆に、オーバーレイビジュアルを入力として与えて、特定の条件を満たす最適なアンダーレイビジュアルを選択することもできます。オーバーレイに関係のない X エラーは別として、選択されたビジュアルを使用すれば、ウィンドウを安全に作成することができます。

このルーチンは、特定のビジュアルの最適なパートナを探索し、条件に合致するビジュアルが見つかったかどうかに応じて成功または失敗を表す状態を戻します。条件には次の 2 つの種類があります。

1. ハード条件 – 必ず満たされなければならない条件。ハード条件を満たすビジュアルだけが最適なペアの候補となります。
2. ソフト条件 – 必須ではないが満たされたほうが望ましい条件。

すべてのハード条件と大部分のソフト条件を満たすビジュアルが選択され、そのビジュアルの属性が戻されます。すべてのハード条件および同じ数のソフト条件を満たすビジュアルが複数見つかった場合は、そのうち 1 つのビジュアルが選択されて戻されます。どのビジュアルが選択されるかは、実装状態によって異なります。

XSolarisOvlSelectPartner の構文と引数を次に示します。

```
XSolarisOvlSelectStatus

XSolarisOvlSelectPartner (Display *display, int screen,   VisualID vid,

XSolarisOvlSelectType seltype, int numCriteria,   XSolarisOvlVisualCriteria

*pCriteria,   XVisualInfo *visinfoReturn,   unsigned long

*unmetCriteriaReturn)
```

display	X サーバーへの接続を指定する。
screen	ビジュアル vid のスクリーンを指定する整数。
vid	パートナを探すビジュアルの XID。
seltype	選択対象となるビジュアルの種類を指定する。

numCriteria	pCriteria 配列の中にあるXSolarisOvlVisualCriteria 構造体の数。
pCriteria	ビジュアルを選択するための条件を優先順位の高い順に指定した XSolarisOvlVisualCriteria 構造体の配列。
visinfoReturn	呼び出し側が提供する XVisualInfo 構造体へのポインタ。正常終了の場合は、選択されたビジュアルの属性がこの構造体に書き込まれる。
unmetCriteriaReturn	満足されなかった条件を記述するビットマスクへのポインタ。この出力引数が意味を持つのは、ルーチンが XSolarisOvlQualifiedSuccessまたは XSolarisOvlCriteriaFailure という値を戻すときに限られる。

引数の型

XSolarisOvlSelectType は、XSolarisOvlSelectPartner 内で選択できる 2 種類のビジュアルを定義する列挙型です。この構造体は、次のように定義されます。

```
typedef

enum {  XSolarisOvlSelectBestOverlay,  XSolarisOvlSelectBestUnderlay, }

XSolarisOvlSelectType;
```

XSolarisOvlVisualCriteria はビジュアルの選択時に使用される各種条件と、それらの条件の重要度を定義する構造体です。この構造体は次のように定義されます。

```
typedef

struct {  unsigned long     hardCriteriaMask;  unsigned

long     softCriteriaMask  int     c_class;  unsigned int     depth;  unsigned

int     minColors;  unsigned int     minRed;  unsigned int     minGreen;

 unsigned int     minBlue;  unsigned int     minBitsPerRGB;  unsigned

int     minBuffers; }

XSolarisOvlVisualCriteria;
```

hardCriteriaMask と softCriteriaMask は、次に示す任意のビットマスクの論理和を値とするビットマスクです。

```
#define

XSolarisOvlVisualClass          (1L<<0) #define
```

```
XSolarisOvlDepth          (1L<<1) #define

XSolarisOvl   MinColors       (1L<<2) #define

XSolarisOvlMinRed          (1L<<3) #define

XSolarisOvl  MinGreen        (1L<<4) #define

XSolarisOvl   MinBlue       (1L<<5) #define

XSolarisOvlMinBitsPerRGB          (1L<<6) #define

XSolarisOvl   MinBuffers       (1L<<7) #define

XSolarisOvlUnsharedPixels          (1L<<8) #define

XSolarisOvlUnsharedColors          (1L<<9) #define

XSolarisOvlPreferredPartner          (1L<<10)
```

戻り値の型

XSolarisOvlSelectStatus はルーチンがビジュアルの探索に成功したかどうか、失敗した場合はその理由を示す値です。戻り値は、次のいずれかになります。

```
typedef

enum {  XSolarisOvlSuccess,  XSolarisOvlQualifiedSuccess,

 XSolarisOvlCriteriaFailure,  XSolarisOvlFailure, }

XSolarisOvlSelectStatus;
```

- 1 つの XSolarisOvlVisualCriteria 構造体で指定された、すべてのハード条件とソフト条件を満たすビジュアルが正常に見つかった場合は、XSolarisOvlSuccess が戻されます。
- 選択されたビジュアルが、1 つの XSolarisOvlVisualCriteria 構造体で指定された、すべてのハード条件は満たすが、ソフト条件は満たさない場合は、XSolarisOvlQualifiedSuccess が戻されます。この場合は、満たされなかったソフト条件の論理和が unmetCriteriaReturn に書き込まれます。
- XSolarisOvlCriteriaFailure は、XSolarisOvlVisualCriteria 構造体で指定されたハード条件を満たすビジュアルが見つからなかったことを示します。この場合は、もっとも緩いハード条件を持つ XSolarisOvlVisualCriteria 構造体について、満たされなかったハード条件の論理和が unmetCriteriaReturn に書き込まれます。
- 条件の不一致以外のエラーが発生した場合は、XSolarisOvlFailure が戻されます。

複数の条件集合

このルーチンでは、優先順位の高い順に条件集合を指定することができます。すなわち、単一の呼び出しで複数の条件集合が指定できます。まず、最初の条件集合に合致するビジュアルの探索がルーチンにより行われ、最初の条件集合のすべてのハード条件を満たすビジュアルが見つかると、そのビジュアルが選択されます。そのようなビジュアルが見つからない場合は、2 番目の条件集合を使用して探索が続けられます。このような手続きは、特定の条件集合のすべてのハード条件を満たすビジュアルが見つかるか、すべての条件集合のテストが終了するまで続けられます。すなわち、この優先順位の仕組みでは、もっとも望ましいビジュアルを最初の条件集合として指定し、それより優先度の低いビジュアルについては 2 番目以降の条件集合として指定すればよいわけです。この仕組みにより、単一のサブルーチン呼び出しで、ビジュアルに関するユーザーの要求を優先順位の高い順に探索することができます。

特定の条件集合では、任意の条件をハード条件またはソフト条件として指定することができます。特定の条件の `hardCriteriaMask` は、探索時にハード条件として指定する条件ビットマスクの論理和です。同様に、`softCriteriaMask` はソフト条件ビットマスクの論理和です。

条件の中には特定の値を取るものがあります。これらの値は、`XSolarisOvlVisualCriteria` 構造体の他のデータメンバによって提供されます。以下の条件の説明では、これらのデータメンバを必要に応じて取り上げます。

- `XSolarisOvlVisualClass` は、選択されたビジュアルが特定のビジュアルクラスを持つことを指定します。そのビジュアルクラスは `c_class` で指定します。
- `XSolarisOvlDepth`, `XSolarisOvlMinColors`, `XSolarisOvlMinRed`, `XSolarisOvlMinGreen`, `XSolarisOvlMinBlue` の各条件は、相互に関連があります。通常は、これらの条件の一部だけを指定します。
- `XSolarisOvlDepth` は、選択されたビジュアルのデプスが `depth` に等しくなることを指定します。
- `XSolarisOvlMinColors` は、選択されたビジュアルで表示可能なカラーの総数が少なくとも `minColors` 個であることを指定します。
- `XSolarisOvlMinRed`、`XSolarisOvlMinGreen`、`XSolarisOvlMinBlue` を使用すれば、`DirectColor` ビジュアルまたは `TrueColor` ビジュアルについて詳細なカラー条件を指定することができます。各条件に対応する値はそれぞれ、`minRed`、`minGreen`、`minBlue` で指定します。これらのデータメンバの値は、選択されたビジュアルの赤、緑、青の要素値の下限を指定するものです。

- XSolarisOvlMinBitsPerRGB は、選択されたビジュアル上で作成されるカラーマップからのカラーチャネル出力が少なくとも minBitsPerRGB であることを指定します。
- XSolarisOvlMinBuffers は、選択されたビジュアルに少なくとも minBuffers 個の高速 MBX イメージバッファを割り当てることを指定します。
- XSolarisOvlUnsharedPixels は、引数のビジュアル vid のウィンドウピクセルとは異なる描画プレーングループ内に存在するウィンドウピクセルを持つパートナビジュアルを選択します。この条件下では、引数のビジュアルと同じ描画プレーングループを使用するビジュアルは選択されません。
- XSolarisOvlUnsharedColors は、オーバーレイ/アンダーレイのウィンドウペアがカラーマップのフォーカスを保持しているときに、ウィンドウピクセルのカラーを同時に表示できるようなパートナビジュアルを選択します。この条件下では、ビジュアルが単一のカラー LUT プールを共有し、そのプール内に引数のビジュアルと同じカラー LUT が 1 個しか存在しない場合、そのビジュアルは選択されません。

条件集合の hardCriteriaMask をゼロに設定すると、任意のビジュアルがその条件集合のハード条件を満たすことになります。同様に、条件集合の softCriteriaMask をゼロに設定すると、その条件集合のソフト条件は必ず満たされることになります。

オーバーレイ/アンダーレイビジュアルの最適なペアの選択

このルーチンは XSolarisOvlSelectPartner と似ていますが、特定のビジュアルのパートナビジュアルを見つけるのではなく、特定のスクリーン上のすべてのビジュアルペアから、指定された条件にもっとも合致するオーバーレイとアンダーレイのペアを同時に選択します。オーバーレイに関係のない X エラーは別として、選択されたビジュアルを使用すれば、ウィンドウを安全に作成することができます。

このルーチンは、pCriteria で指定された条件に基づいて、特定のスクリーンの最適なビジュアルペアを探索し、それからすべてのビジュアルペア (最適なペアとそれ以外のペア) を探索します。オーバーレイとアンダーレイに関する条件は、pCriteria の各要素で指定されます。このルーチンは、指定された条件に合致するペアが見つかったかどうかに応じて成功または失敗を表す状態を戻します。

選択されたペアのオーバーレイビジュアルは、オーバーレイに関するすべてのハード条件を満たします。また、このペアのアンダーレイビジュアルは、アンダーレイ

に関するすべてのハード条件を満たします。オーバーレイビジュアルの属性は ovVisinfoReturn に戻され、アンダーレイビジュアルの属性は unVisinfoReturn に戻されます。すべてのハード条件 (オーバーレイとアンダーレイ) および同じ数のソフト条件 (オーバーレイまたはアンダーレイ) を満たすビジュアルペアが複数見つかった場合は、そのうち 1 つのペアが選択されて戻されます。どのペアが選択されるかは、実装状態によって異なります。

構文と引数を次に示します。

```
XSolarisOvlSelectStatus

XSolarisOvlSelectPair (Display *display, int screen, int numCriteria,

  XSolarisOvlPairCriteria *pCriteria,   XVisualInfo *ovVisinfoReturn,

XVisualInfo *unVisinfoReturn,   unsigned long *unmetOvCriteriaReturn,

  unsigned long *unmetUnCriteriaReturn)
```

display	X サーバーへの接続を指定する。
screen	ビジュアルの探索場所となるスクリーンを指定する整数。
numCriteria	pCriteria 配列の中にある XSolarisOvlPairCriteria 構造体の数。
pCriteria	ビジュアルのペアを選択するための条件を優先順位の高い順から低い順に指定した XSolarisOvlPairCriteria 構造体の配列。
ovVisinfoReturn	呼び出し側が提供する XVisualInfo 構造体へのポインタ。正常終了の場合は、選択されたオーバーレイビジュアルの属性がこの構造体に書き込まれる。
unVisinfoReturn	呼び出し側が提供する XVisualInfo 構造体へのポインタ。正常終了の場合は、選択されたアンダーレイビジュアルの属性がこの構造体に書き込まれる。

unmetOvCriteriaReturn	オーバーレイビジュアルに関して満足されなかった条件を記述するビットマスクへのポインタ。この出力引数が意味を持つのは、ルーチンが XSolarisOvlQualifiedSuccess または XSolarisOvlCriteriaFailure という値を戻すときに限られる。
unmetUnCriteriaReturn	アンダーレイビジュアルに関して満足されなかった条件を記述するビットマスクへのポインタ。この出力引数が意味を持つのは、ルーチンが XSolarisOvlQualifiedSuccess または XSolarisOvlCriteriaFailure という値を戻すときに限られる。

引数の型

XSolarisOvlPairCriteria はビジュアルの選択時に使用される 各種条件と、それらの条件の重要度を定義する構造体です。この構造体は次のように定義されます。

```
typedef

struct {  XSolarisOvlVisualCriteria      overlayCriteria;

 XSolarisOvlVisualCriteria      underlayCriteria; }

XSolarisOvlPairCriteria;
```

XSolarisOvlVisualCriteria は、XSolarisOvlSelectPartner の仕様で定義されています。

戻り値の型

XSolarisOvlSelectStatus はこの型の定義については、XSolarisOvlSelectPartner の仕様を参照してください。

- XSolarisOvlPairCriteria 構造体の 1 つで指定されたすべてのハード条件とソフト条件を満たすビジュアルペアが正常に見つかった場合は、XSolarisOvlSuccess が戻されます。
- 選択されたビジュアルが XSolarisOvlPairCriteria 構造体の 1 つで指定されたすべてのハード条件を満たすが、ソフト条件は満たさない場合は、XSolarisOvlQualifiedSuccess が戻されます。この場合は、オーバーレイとアンダーレイに関して満たされなかったソフト条件の論理和が、それぞれ unmetOvCriteriaReturn と unmetUnCriteriaReturn に書き込まれます。
- XSolarisOvlCriteriaFailure は、XSolarisOvlPairCriteria 構造体で指定されたハード条件を満たすビジュアルペアが見つからなかったことを示します。この場合は、オーバーレイとアンダーレイに関

し、XSolarisOvlPairCriteria 構造体の満たされなかったハード条件のうち、もっとも緩いハード条件の 論理和が、それぞれ unmetOvCriteriaReturn と unmetUnCriteriaReturn に書き込まれます。

- 条件の不一致以外のエラーが発生した場合は、XSolarisOvlFailure が戻されます。

条件集合

XSolarisOvlSelectPartner と同様に、XSolarisOvlSelectPair では、優先順位の高い順に条件集合を指定することができます。すなわち、単一の呼び出しで複数の条件集合が指定できます。まず、オーバーレイとアンダーレイに関する最初の条件集合に合致するビジュアルペアの探索がルーチンにより行われ、最初の条件集合のすべてのハード条件を満たすペアが見つかると、そのペアが選択されます。そのようなペアが見つからない場合は、2 番目の条件集合を使用して探索が続けられます。このような手続きは、特定の条件集合のすべてのハード条件を満たすペアが見つかるか、すべての条件集合がテストされるまで続けられます。すなわち、この優先順位の仕組みでは、もっとも望ましいペアを最初の条件集合として指定し、それより優先度の低いペアについては 2 番目以降の条件集合として指定すればよいわけです。この仕組みにより、単一のサブルーチン呼び出しで、ペアに関するユーザーの要求を優先順位の高い順に探索することができます。

指定できる条件マスクの詳細については、93ページの「オーバーレイ/アンダーレイウィンドウのビジュアルの選択」を参照してください。

第 7 章

セキュリティ

Solaris 環境は、ユーザーベースとホストベースの 2 種類のアクセス制御機構をサポートするとともに、MIT-MAGIC-COOKIE-1 および SUN-DES-1 という 2 種類の認証プロトコルをサポートしています。この章では、これらのアクセス制御機構と認証プロトコルについて解説するほか、サーバーのアクセス制御を変更する方法およびクライアントの実行方法について説明します。その実行方法には、リモートで実行する方法と別のユーザーとしてローカルで実行する方法の 2 種類があります。

この章についての注意

この章の内容が問題となるのは、以下に示すいずれかのコンフィギュレーションでアプリケーションを実行する場合です。

- OpenWindows 2.0 または X11R4 より前のバージョンの `Xlib` とリンクされたアプリケーションを実行する。詳細は、105ページの「ホストベース」を参照してください。
- OpenWindows 2.0 ライブラリに静的にリンクされたアプリケーションを実行し、かつ SUN-DES-1 認証プロトコルを適用しようとする。詳細は、106ページの「SUN-DES-1」を参照してください。
- リモートサーバー上でアプリケーションを実行する。詳細は、111ページの「クライアントをリモートで実行する場合とローカルで実行する場合」を参照してください。

上記のコンフィギュレーションのいずれも使用しない場合は、デフォルトのセキュリティの設定を変更する必要はありません。

アクセス制御機構

どのクライアント (すなわちアプリケーション) が OpenWindows サーバーにアクセスできるかを制御するのがアクセス制御機構です。アクセスが許可されたクライアントだけがサーバーに接続でき、許可されていない X クライアントはすべて以下のエラーメッセージが表示され終了します。

```
Xlib:

connection to hostname refused by server Xlib:

Client is not authorized to connect to

server
```

サーバーコンソールに以下のメッセージが表示されます。

```
AUDIT:

<月日 時刻 年>: X: client

6 rejected from IP

129.144.152.193 port

3485  Auth name:

MIT-MAGIC-COOKIE-1
```

アクセス制御機構には、ユーザーベースとホストベースの 2 種類があります。openwin とともに -noauth オプションを指定しなければ、その両方のアクセス制御機構がアクティブ状態となります。詳細は後述の 107ページの「サーバーに対するアクセスの操作」を参照してください。

ユーザーベース

ユーザーベース、すなわち認証ベースの制御機構では、あるユーザーに対し、任意のホスト上の特定ユーザーに対するアクセスが明示的に許可されます。すなわち、そのユーザーのクライアントがサーバーに認証データを渡し、そのデータがサーバーの認証データと一致すれば、アクセスが許可されます。

ホストベース

汎用機構としてのホストベース制御機構では、ユーザーに対して特定ホストへのアクセスが許可され、そのホスト上の全ユーザーがサーバーに接続できます。これはゆるやかなアクセス制御方式であり、サーバーに対するアクセス権がホストに与えられていれば、そのホスト上の全ユーザーがサーバーに接続できます。

ホストベース制御機構は、主に旧バージョンとの互換性を目的として使用されます。OpenWindows 2.0 または X11R4 より前のバージョンの `Xlib` とリンクされたアプリケーションは、新しいユーザーベースのアクセス制御機構に対応できません。それらのアプリケーションがサーバーに接続できるようにするためには、ホストベース機構に切り換えるか、あるいは新バージョンの `Xlib` とリンクし直さなければなりません。

注 - 旧バージョンの Xlib とリンクしたクライアントは、できれば新しいバージョンとリンクし直してください。これにより、新しいユーザーベースのアクセス制御機構の下でサーバーに接続できるようになります。

認証プロトコル

OpenWindows 環境では、MIT-MAGIC-COOKIE-1 および SUN-DES-1 の 2 種類の認証プロトコルをサポートしています。その 2 種類のプロトコルの違いは、使用する認証データの違いであり、アクセス制御機構に関しては同様です。

ユーザーベース機構を使用する MIT-MAGIC-COOKIE-1 プロトコルが OpenWindows 環境のデフォルト設定です。

MIT-MAGIC-COOKIE-1

MIT-MAGIC-COOKIE-1 は、MIT (マサチューセッツ工科大学) で開発された認証プロトコルです。マジッククッキーは、ランダムな値に生成されるバイナリ形式のパスワードです。サーバーのスタートアップ時に、そのサーバーおよびユーザー (システムを起動したユーザー) に関してマジッククッキーが作成されます。接続が要求されるたびに、そのユーザーのクライアントは、接続パケットの一部として、サーバーにマジッククッキーを送ります。そのマジッククッキーはサーバーのマジッククッキーと比較され、2 つが一致すれば接続が許可されます。2 つのマジッククッキーが一致しなければ、接続は許可されません。

SUN-DES-1

SUN-DES-1 は Sun が開発した認証プロトコルです。Secure RPC (Remote Procedure Call) が基本になっており、DES (Data Encryption Software) サポートが必要です。認証データはユーザーのネット名、すなわちマシン独立のネットワーク名です。このデータは暗号化され、接続パケットの一部としてサーバーに送られます。サーバーはそのデータを暗号解読し、それが既知のネット名であれば接続が許可されます。

SUN-DES-1 認証プロトコルは、MIT-MAGIC-COOKIE-1 よりも高度なセキュリティを提供します。特定ユーザーのネット名 (マシン独立) を他のユーザーが使用してサーバーにアクセスすることは不可能です。しかし、他のユーザーがマジッククッキーを使用してサーバーにアクセスすることは可能です。

このプロトコルは、OpenWindows バージョン 3 以降の環境に入っているライブラリでしか使用できません。それより古い OpenWindows 環境の静的ライブラリ (特に Xlib) とリンクしているアプリケーションは、この認証プロトコルを使用できません。

後述の 110ページの「SUN-DES-1 を使用する場合のアクセス許可」で説明するように、特定ユーザーのアクセスリストにネット名を追加することによって、サーバーに対する他ユーザーのアクセスを許可することができます。

デフォルト認証プロトコルの変更

デフォルト認証プロトコルの MIT-MAGIC-COOKIE-1 を他のサポートされている認証プロトコルに変更することができます。また、ユーザーベースのアクセス制御機構を使用しないという変更も可能です。このデフォルト設定の変更には、`openwin` コマンドとともにオプションを指定します。詳しくは `openwin(1)`のマニュアルページを参照してください。

たとえば、MIT-MAGIC-COOKIE-1 から SUN-DES-1 に変更するには、OpenWindows を次のように起動します。

```
example%
openwin -auth
sun-des
```

ユーザーベースのアクセス機構なしで OpenWindows を実行する必要があるときは、コマンド行オプション `-noauth` を使用します。

```
example%
openwin -noauth
```

注意 - -noauth を使用することによってセキュリティ機能が低下します。ホストベースのアクセス制御機構のみで OpenWindows を実行することに等しく、サーバーはユーザーベースのアクセス制御機構を非アクティブ状態とします。そのローカルマシン上でアプリケーションを実行できるすべてのユーザーに対してサーバーへのアクセスが許可されます。

サーバーに対するアクセスの操作

openwin コマンドで -noauth オプションを指定しなければ (106ページの「デフォルト認証プロトコルの変更」を参照)、ユーザーベースとホストベースの両方のアクセス制御機構がアクティブ状態となります。サーバーは、最初にユーザーベース機構を調べ、次にホストベース機構を調べます。デフォルトのセキュリティコンフィギュレーションでは、ユーザーベース機構として MIT-MAGIC-COOKIE-1、ホストベース機構に対して空リストが使用されます。ホストベースのリストが空であるため、アクティブになるのはユーザーベース機構です。-noauth オプションを指定すると、サーバーはユーザーベースのアクセス制御機構をアクティブでない状態とします。また、ローカルホストを追加することによってホストのベースリストが初期化されます。

サーバーのアクセス制御機構を変更するためのプログラムとして、xhostとxauthの 2 つがあります。詳しくは xhost および xauth のマニュアルページを参照してください。これらのプログラムは、認証プロトコルによって作成された 2 つのバイナリファイルにアクセスします。これは、特定セッションの認証データを格納したファイルです。一方のファイルはサーバーが内部的に使用するためのもので、もう一方のファイルは下記のファイル名でユーザーの $HOME ディレクトリに配置されます。

表 7–1

.Xauthority	(クライアント認証ファイル)

サーバー中のホストベースアクセスリストを変更するには、xhost プログラムを使用します。これにより、アクセスリストに対するホストの追加および削除が可能です。最初にデフォルトのコンフィギュレーションである空のホストベースアクセス

リストを使用し、xhost によってマシン名を追加するという方法は、セキュリティレベルを低下させます。すなわち、追加されたホストがアクセスを許可されるだけでなく、デフォルト認証プロトコルを指定するすべてのユーザーもアクセスが許可されます。ホストベースアクセス制御機構のセキュリティレベルが低いと見なされる理由は前述の 105ページの「ホストベース」の項で説明した通りです。

xauth プログラムは、.Xauthority クライアントのファイル中の認証プロトコルデータにアクセスします。アクセスを操作する側のユーザーは自分の .Xauthority ファイルからデータを抽出して、他のユーザーが自分の .Xauthority ファイルにそのデータを併合できるようにすることができます。これにより、操作側ユーザーのサーバーまたはそのユーザーが接続しているサーバーに対して他のユーザーがアクセス可能になります。

xhost および xauth の使用例は後述の109ページの「MIT-MAGIC-COOKIE-1 を使用する場合のアクセス許可」に示してあります。

クライアント認証ファイル

クライアント認証ファイル .Xauthority のエントリには以下の形式があります。

表 7–2

connection-protocol	*auth-protocol*	*auth-data*

デフォルト設定では、.Xauthority に *auth-protocol* として MIT-MAGIC-COOKIE-1 が格納され、ローカル表示に関するエントリが *connection-protocol* および *auth-data* としてのみ格納されます。たとえば、ホスト *anyhost* 上で .Xauthority ファイルに以下のエントリが格納されているという場合が考えられます。

表 7–3

anyhost:0	MIT-MAGIC-COOKIE-1	*82744f2c4850b03fce7ae47176e75*
localhost:0	MIT-MAGIC-COOKIE-1	*82744f2c4850b03fce7ae47176e75*
anyhost/unix:0	MIT-MAGIC-COOKIE-1	*82744f2c4850b03fce7ae47176e75*

クライアントのスタートアップ時に、*connection-protocol* に対応するエントリが `.Xauthority` から読み取られ、*auth-protocol* および *auth-data* が接続パケットの一部としてサーバーに送られます。デフォルトのコンフィギュレーションでは、`xhost` によって空のホストベースアクセスリストが戻され、認証が実行可能な状態であることを示します。

認証プロトコルをデフォルトから SUN-DES-1 に変更した場合、`.Xauthority` には *auth-protocol* として SUN-DES-1 が格納され、*auth-data* としてユーザーのネット名が格納されます。ネット名の形式は次の通りです。

unix.*userid@NISdomainname*

たとえば、ホスト *anyhost* 上で、`.Xauthority` ファイルに以下のエントリが格納されているという場合が考えられます。

表 7–4

anyhost:0	SUN-DES-1	"unix.*15339@EBB.Eng.Sun.COM*"
localhost:0	SUN-DES-1	"unix.*15339@EBB.Eng.Sun.COM*"
anyhost/unix:0	SUN-DES-1	"unix.*15339@EBB.Eng.Sun.COM*"

この unix.*15339@EBB.Eng.Sun.COM* は、マシン独立であるユーザーのネット名です。

注 - 自分のネットワーク名、またはマシン独立のネット名が不明のときは、システム管理者に照会してください。

MIT-MAGIC-COOKIE-1 を使用する場合のアクセス許可

MIT-MAGIC-COOKIE-1 認証プロトコルを使用している場合、次の手順により、自分のサーバーに他のユーザーがアクセスできるようにすることができます。

1. サーバーを走らせているマシン上で `xauth` を実行し、***hostname: 0*** に対応するエントリを抽出してファイルに入れる。

   例として、*hostname* が *anyhost*、ファイルが *xauth.info* である場合を示します。

```
myhost%$OPENWINHOME/bin/xauth

   nextract - anyhost:0 >

   $HOME/xauth.info
```

2. エントリが入っているファイルを、アクセスを要求しているユーザーに送る (メールツール、rcp またはその他のファイル転送プロトコルを使用する)。

注 - 認証情報が格納されたファイルの転送には、rcp よりもメールを使用した方がより安全です。rcp を使用する場合は、他のユーザーが容易にアクセスできるディレクトリにそのファイルを入れないようにしなければなりません。

---

3. 他のユーザーはエントリを自分の .Xauthority ファイルに併合しなければならない。

   例として、*userhost* が xauth.info を自分の .Xauthority ファイルに併合する場合を示します。

```
userhost%
   $OPENWINHOME/bin/xauth nmerge - <
   xauth.info
```

注 - *auth-data* は特定セッションに対応します。したがって、それが有効とされるのはサーバーが再起動されるまでの間です。

SUN-DES-1 を使用する場合のアクセス許可

SUN-DES-1 認証プロトコルを使用している場合、次の手順により、自分のサーバーに他のユーザーがアクセスできるようにすることができます。

1. サーバーを走らせているマシン上で xhost を実行し、サーバーに新しいユーザーを通知する。

   例として、新しいユーザー *somebody* に *myhost* 上での実行を許可する場合を示します。

```
myhost%
   xhost +
   somebody@
```

2. 新しいユーザーは、xauth を使用してエントリを自分の .Xauthority ファイルに追加しなければならない。

   例として、新しいユーザー *somebody* のマシン独立のネット名が unix.*15339*@*EBB.Eng.Sun.COM* である場合を示します。

```
userhost%
    echo 'add
    myhost:0
    SUN-DES-1

    ``unix.15339@EBB.Eng.Sun.COM'''

    | $OPENWINHOME/bin/xauth
```

クライアントをリモートで実行する場合とローカルで実行する場合

X クライアントは、接続するサーバーの名前を取り出すために、環境変数 DISPLAY の値を使用します。

クライアントをリモートで実行するか、またはローカルで他ユーザーとして実行する場合の手順は次の通りです。

1. サーバーを走らせているマシン上で、他のユーザーにアクセスを許可する。

   使用するプロトコルに応じて、前述の109ページの「MIT-MAGIC-COOKIE-1 を使用する場合のアクセス許可」または110ページの「SUN-DES-1 を使用する場合のアクセス許可」で説明した手順を実行します。

2. DISPLAY の値として、サーバーを実行しているホストの名前に設定する。

   例として、ホストが *remotehost* である場合を示します。

```
myhost%
    setenv DISPLAY
    remotehost:0
```

3. クライアントプログラムを実行する。

   クライアントはリモートマシン *remotehost* 上に表示されます。

```
myhost%
   client_program&
```

付録 **A**

基準表示デバイス

この付録では、Solaris 基準表示デバイスと、それによってエクスポートされるビジュアルについて説明します。ビジュアルについての詳細は、第 3 章を参照してください。

Solaris 基準表示デバイス

Solaris 環境では、特定の表示デバイスが基準デバイスと見なされます。このようなデバイスには、Solaris Device Developer Kit (DDK) で提供されるサンプルデバイスハンドラが付いています。基準デバイスハンドラのコード例をテンプレートとして使用し、独自のデバイスハンドラを作成できます。

デバイスハンドラを作成して構成するプロセスについては、『*X Server Device Developer's Guide*』を参照してください。Solaris X サーバーでは、有効なデバイスハンドラが作成され、システムに対して構成されているデバイスについてはすべてサポートしています。

Solaris 基準デバイスとビジュアル

表 A–1 は、基準表示デバイスとそれによってエクスポートされるビジュアルを示します。デバイス名はサーバーに対してディスプレイアダプタを指定し、製品名はディスプレイカードのタイプを指定します。1 つのデバイスに別の製品名が付いている場合は、その製品名が CG*n* デバイス名の代わりに使用されているので注意してください (たとえば、CG8 ではなく TC が使用されている)。

エクスポートされるデプスは、この特殊なデバイスタイプの画面用にサーバーによって広告 (advertise) されるビジュアルのデプスを指定します。MPG (複数プレーングループ) は、デバイスで複数デプスのビジュアルがサポートされることを示します。この表で使用している用語についての詳細は、用語集を参照してください。

表 A–1 Solaris 基準表示デバイス

デバイス名	製品名	デバイスドライバ	バス	エクスポートされるデプス
BW2	なし	`/dev/fbs/bwtwo`*X*	SBus, VME/obio, P4	1 ビット
CG3	なし	`/dev/fbs/cgthree`*X*	SBus	8 ビット
CG6	GX	`/dev/fbs/cgsix`*X*	SBus, P4	8 ビット
CG6	GXplus/TurboGXplus	`/dev/fbs/cgsix`*X*	SBus	8 ビット
CG8	TC	`/dev/fbs/cgeight`*X*	SBus, P4	1 ビット、24 ビット (MPG)
leo	LEO	`/dev/fbs/leo0`	SBus	1 ビット、24 ビット (MPG)
ffb	FFB	`/dev/fbs/ffb0`	SBus	1 ビット、24 ビット (MPG)
m64	PGX	`/dev/fbs/m64`*X*	PCI	8 ビット
vga4	VGA	該当なし	ISA, EISA, MCA	8 ビット
vga8	VGA	該当なし	ISA, EISA, MCA	8 ビット
i8514	8514/A	該当なし	ISA, EISA, MCAS	8 ビット

注 - サーバーは、最大 16 のディスプレイをサポートするように構成されています。制限があるとすれば、ハードウェアでサポートされるフレームバッファ数によるものです。

SPARC: SPARC でサポートされる基準デバイス

BW2

BW2 は、モノクロモニタをサポートする単純な 1 ビットのフレームバッファです。このデバイスのデバイスハンドラは、1 ビットの StaticGray ビジュアルのみをエクスポートします。したがって、これは組み込みデフォルトビジュアルです。サードパーティ製品を含む各種のバスと画面解像度に合わせて、さまざまな BW2 フレームバッファが販売されています。

CG3

CG3 は、SBus システム用の 8 ビットインデックスカラーのダムフレームバッファです。このデバイスのデバイスハンドラは、複数の 8 ビットビジュアル (後述) をエクスポートします。組み込みデフォルトビジュアルは 8 ビット PseudoColor です。

GX ファミリーのデバイス

GX は、2 次元と 3 次元のワイヤフレーム、均一にシェーディングされたポリゴン、全般的なウィンドウシステムの高速化を目的とする 8 ビットインデックスカラー表示アクセラレータです。ウィンドウシステムの高速化は自動的に実行され、その他の高速化機能は Solaris のグラフィックス API を介して指定できます。複数の 8 ビットビジュアルがサポートされ、組み込みデフォルトビジュアルは 8 ビット PseudoColor です。GX は SBus と P4 バス用として提供されます。

GXPlus デバイスは、基本的には SBus システム上のダブルバッファリング機能と画面解像度向上のために使用できるメモリーを GX に追加したようなものです。この GXplus を使用できる場合、SolarisX サーバーは見えない画面領域を使用して X11 ピックスマップを自動的に高速化します。

TC (CG8)

TC デバイスは、2 種類のメモリーバッファ、つまり、1 ビットモノクロと 24 ビットカラーというプレーングループを保有しています。どちらのプレーングループ内でもウィンドウを作成できるので、MPG デバイスです。1 ビットと 24 ビットのすべてのビジュアルがサポートされます。

一部の (以前の) X11 クライアントアプリケーションの中には、カラーフレームバッファが 8 ビットの組み込みデフォルトビジュアルを使用するものと見なし、TC 上でカラー表示しないものがあります。このような結果を防ぐため、組み込みデフォルトビジュアルは 1 ビットの StaticGray です。

TC のプレーングループは、完全に別々なメモリーバッファになっているため、相互に衝突することはありません。OpenWindows では、デフォルト時にこれを利用することにより、1 ビットのウィンドウと 24 ビットのウィンドウが互いに干渉し合わないようにして、システムの性能を向上させます。この動作を「最小の露出」といいます。この動作は、`openwin(1)` の `-nominexp` オプションを使用すれば無効にできます。このオプションを使用すると、1 ビットのウィンドウと 24 ビットのウィンドウは互いに干渉し合うようになります。

Solaris X サーバーは、可能であれば他の MPG デバイスにも最小のエクスポージャを適用します。これらのデバイスでは、`openwin` の `-nominexp` オプションを使用してください。

注 - X プロトコルは、任意にカーソルの構成要素を変形させることができます。システムの性能を全般的に向上させるため、OpenWindows サーバーは常にカーソルを TC の 1 ビットのプレーングループにして描画します。

IA: IA でサポートされる基準デバイス

VGA

VGA はカラーのダムフレームバッファです。サーバーは、すべてのビジュアル型とデフォルトの PseudoColor (vga8) を備えた 8 ビットのインデックスカラー、または 4 ビットの StaticColor (vga4) として VGA をサポートします。8 ビットモードを使用する場合、通常の解像度は 1024 x 768 になります。4 ビットモードはすべての VGA デバイス上で利用できる基本的な VGA グラフィックスモードなので、通常の解像度は 640 x 480 に制限されます。大部分の VGA の bitsPerRGB は 6 です。vga8 サーバーにも、XGA をダムフレームバッファとしてサポートする機能があります。

4 ビットの VGA モードでは、VGA パンニングがサポートされます。パンニングモードでは、より大きい仮想ディスプレイに物理ウィンドウをマップできます。仮想ディスプレイ内で移動する場合は、画面の縁の外側にマウスを「押し出して」ください。物理ウィンドウが仮想ディスプレイの境界に当たるまで、仮想ディスプレイ内の物理ウィンドウがマウスの押された方向に自動的に移動します。

OpenWindows を十分に使い慣れたユーザー以外は、パンニングを使用しないでください。メッセージによる通知なしで、アイコンやポップアップボックス (メニュー / ダイアログボックスなど) が画面の外側に表示される場合があります。このような状況を把握し、隠れたウィンドウオブジェクトを探して通常の状態に回復できる程度の使用経験が必要です。パンニングを使用するときは、できるだけポップアップポインタをジャンプさせるようにしてください。`olvwm` や `tvwm` などの仮想ウィンドウマネージャを使用すると混乱する危険性が大きいため、これらは使用しないでください。

8514/A

8514/A は、ウィンドウシステムを全体的に高速化する 8 ビットのインデックスカラーグラフィックスアクセラレータです。このデバイスを使用すると、VGA に比べて性能を大幅に改善できます。サーバーで 8514/A を使用する場合は、8 ビットのインデックスカラーで、解像度は 1024x768 または 1280x1024 に制限されます。サーバーでは 8 ビットのすべてのビジュアルがサポートされます。組み込みビジュアルは 8 ビットの PseudoColor です。ほとんどの 8514/A アクセラレータの bitsPerRGB は 6 です。

用語集

RGB
R、G、B はそれぞれ赤、緑、青のモニターガンを駆動する電圧レベルです。

アウトラインフォント
アウトラインフォントとは、理想的な形状をした文字の集まりです。各形状は、形状の内部と外部とを分離する連続曲線の線分によって数値的に定義されます。この方法は写真植字機のような高解像度デバイスで使用されています。

アクセス制御機構
アクセス制御機構とは、どのクライアントまたはアプリケーションが OpenWindows サーバーにアクセスできるかを決定する手段です。アクセス制御機構には、ユーザーベースとホストベースの 2 種類あります。

ウィンドウ
ウィンドウはクライアントがテキストとグラフィックスを描く描画面となります。1 つのクライアントは複数のウィンドウを使用できます。

ウィンドウサーバー
Solaris X サーバーのようなウィンドウサーバーまたはティスプレイサーバーは、マシンの表示機能を処理し、ユーザーデバイスと他のクライアントからの入力を集め、イベントをクライアントに送信するプログラムです。Solaris X サーバーはウィンドウマネージャとのすべての通信を処理します。

ウィンドウマネージャ
スクリーン上でのウィンドウの操作とユーザーインタフェース (方針) の大半は一般に、ウィンドウマネージャクライアントによって提供されます。ウィンドウマネージャは ウィンドウサーバーとのみ通信します。

ウィンドウ **ID** テーブル記述子	ウィンドウ ID (WID) テーブルにはピクセルのビジュアル面、例えば、8 ビットピクセルであるか 24 ビットピクセルであるか、ピクセルを表示するときにはどの ルックアップテーブル (LUT) を使用すべきか、ピクセルがダブルバッファされているか、といった記述が含まれています。
拡張	システムの機能性を拡張するためにコアプロトコルに対する拡張を定義できます。
仮想カラーマップ	ハードウェアカラー LUT にインストールされるまで不可視であるソフトウェアカラーマップです。
カラールックアップテーブル	カラールックアップテーブルとは、ピクセル値と RGB カラー値の間のマッピングを提供するハードウェアデバイスです。ルックアップテーブル (LUT) とも呼ばれます。
カラーマップフラッシング	クライアントのカラーマップはある時点で 1 つだけインストールされます。インストールされたカラーマップに関連づけられたウィンドウには、その正しいカラーが表示されます。その他のカラーマップに関連づけられたウィンドウには、誤ったカラーが表示されます。この誤ったカラーの表示はカラーマップフラッシングと呼ばれます。
クライアント	クライアントとは、あるプロセス間通信によってウィンドウサーバーに接続されるアプリケーションプログラムです。ウィンドウサーバーのクライアントと呼ばれます。クライアントはウィンドウサーバーと同じマシンで実行でき、またはネットワークの別のマシンで実行されているサーバーに接続できます。Solaris X サーバーのクライアントは X11 プロトコルを介して通信しなければなりません。
クライアントサーバーモデル	分散アプリケーションを作成するときにもっとも一般的に使用されるパラダイムは、クライアントサーバーモデルです。このモデルでは、クライアントはウィンドウサーバープロセスにサービスを要求します。クライアントとサーバーは、接続の両端で実現されなければならないプロトコルを必要とします。Solaris X サーバーは X11 プロトコルを実現します。

グラフィックスアクセラレータ
: イメージをフレームバッファに描く速度を高めるために、回路に組み込まれる表示デバイスのことをアクセラレータまたはグラフィックスアクセラレータと呼びます。グラフィックスアクセラレータにはメモリーおよび追加カラー、3D イメージ、アニメーションの表示など機能性を強化するような回路が含まれることがよくあります。

グラフィックスアダプタ
: 表示デバイスを参照。

サーバー用デフォルトのカラーマップ
: 特別な指定がない限り、サーバーの起動時には PseudoColor ビジュアルが作成されます。このカラーマップはサーバー用デフォルトのカラーマップです。

処理の対象とする **(Express Interest)**
: クライアントがあるイベントについての通知を特に要求したときに、クライアントはそのイベントを処理の対象としたことになります。

スクリーン
: スクリーンとは、物理的なモニターとハードウェアのことで、カラーと白黒があります。一般的な構成は、複数のスクリーン間で１つのキーボードとマウスを共有する形です。

製品名
: 製品名は、ディスプレイカードのタイプを識別します。

接続
: クライアントとサーバーの間の通信経路です。

ソフトウェアカラーマップ
: ソフトウェアカラーマップとは、カラー LUT が提供するカラーマッピングの過程をソフトウェアで抽象化したものです。ソフトウェアカラーマップはハードウェアカラー LUT にロード、またはインストールできます。カラーマップとも呼ばれます。

デバイスドライバ
: デバイスドライバとは、UNIX ファイルシステム内のデバイスの名前です。この場合、*X* は、システム上の特定のデバイスの番号です。たとえば、システムに 2 つの CG3 があれば、第 1 の CG3 の名前は `/dev/fbs/cgthree0` で、第 2 の CGS は `/dev/fbs/cgthree1` となります。システムに CG3 と GX が 1 つずつある場合は、CG3 が `/dev/fbs/cgthree0` となり、GX は `/dev/fbs/cgsix0` となります。

デフォルトビジュアル
: デフォルトビジュアルとは、表示デバイスに対して使用可能なビジュアルの 1 つです。クライアントプログラムを開始すると、プログラムは別のビジュアルを指定しない限り、デフォルトビジュアル

で実行されるのが普通です。デフォルトビジュアルには組み込み、サーバー、許可可能の数種類があります。

ハードウェアカラーマップ　ハードウェアカラーマップとは、カラー LUT です (カラールックアップテーブルも参照)。

バス　バスとは、システムにおける入出力 (I/O) のリンクです。表示デバイスは、バスによってシステムに物理的および論理的に接続されます。SBus、VME、P4 バスは、SPARC システム内で使用されます。サードパーティのシステムでは、この 3 つのバスとは異なるバスが使用されることがあります。

ビジュアル　ビジュアルはピクセル値を解釈する方法を記述します。ビシュアルクラスとピクセルサイズ属性がひとまとめになってビジュアルを記述します。

ビジュアルカテゴリ　ビジュアルカテゴリとは、特定のピクセルサイズのすべてのビジュアルクラスのグループです。Solaris X でサポートされているビジュアルカテゴリは、1 ビット、4 ビット、8 ビット、および 24 ビットです。

ビジュアルクラス　ビジュアルクラスとは、ピクセルをカラーとしてどのように表示するかという方法です。Solaris X でサポートされているビジュアルクラスは、PseudoColor、StaticColor、GrayScale、StaticGray、TrueColor、および DirectColor です。

ピックスマップ　ピックスマップとは、サーバー中のスクリーン以外のメモリーのブロックです。つまり、ピクセル値の配列です。

ビットマップ　ビットマップとは、四角形をした要素の配列であり、各要素には内部値または外部値のどちらかが入ります。

ビットマップフォント　ビットマップフォントとは、ビットマップの集まりとビットマップがどのように使われるかを定義した情報 (例えば、文字の間隔) を合わせたものです。

表示デバイス　ユーザーのモニターは、モニターに表示される内容を制御する表示デバイスに接続されています。表示デバイスには表示情報の保存専用のメモリー (フレームバッファと呼ばれる) も含まれています。表示デバイスはグラフィックスアダプタとも呼ばれます。

複合フォント	複合フォントとは、階層的に構成されたベースフォントの集まりです。
複数デプスデバイス	TC 表示デバイスは、さまざまなデプスのビジュアルを提供するので、複数プレーングループ (MPG) または複数デプスデバイスと呼ばれます。
複数プレーングループ	複数のビジュアルカテゴリを同時にサポートできる表示デバイスは、複数プレーングループ (MPG) デバイスと呼ばれます。
プレーングループ	ピクセルデータが保存されている表示デバイス上の物理メモリーは、一般にプレーングループと呼ばれます。
フレームバッファ	通常、ピクセルデータは、フレームバッファまたはビデオメモリーと呼ばれる専用のコンピュータメモリーに保存されます。
要求	要求とは、接続を介してサーバーに送信されるコマンドです。
ルックアップテーブル	カラールックアップテーブルを参照。

索引