ソフトウェアアークテクチャ
第13回　ウインドウシステム

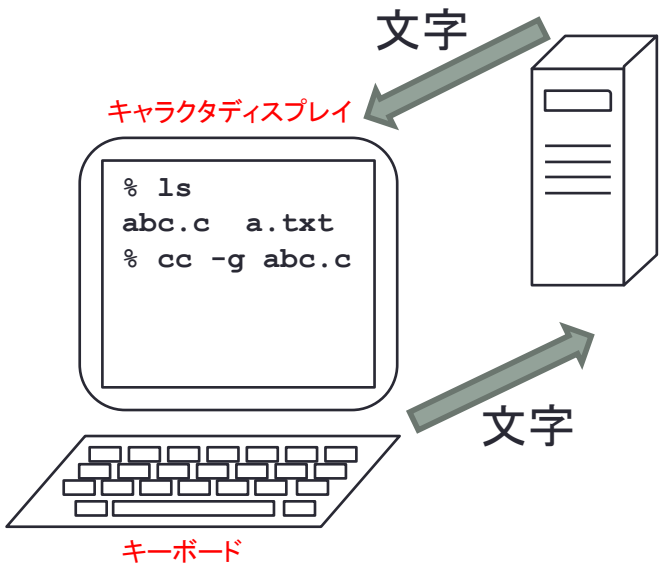
環境情報学部
萩野　達也

スライドURL

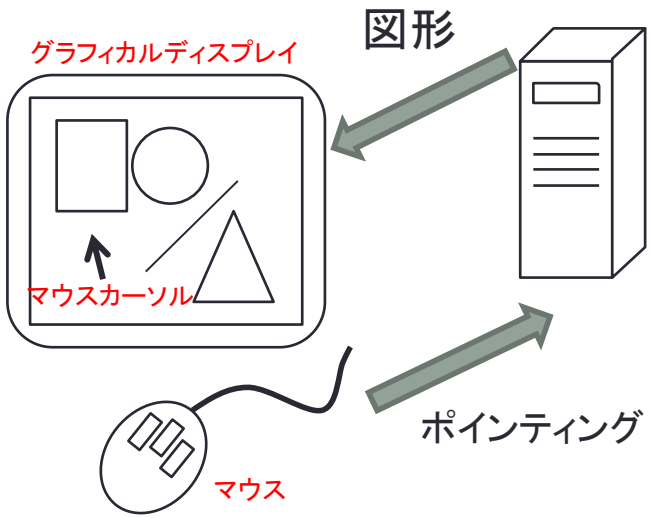
https://vu5.sfc.keio.ac.jp/slide/

CUIからGUIへ

- CUI, CLI, TUI
  - CUI = Character(-based) User Interface, Console User Interface
  - CLI = Command Line Interface
  - TUI = Text User Interface
  - キーボードからの文字入力
  - 端末への文字出力

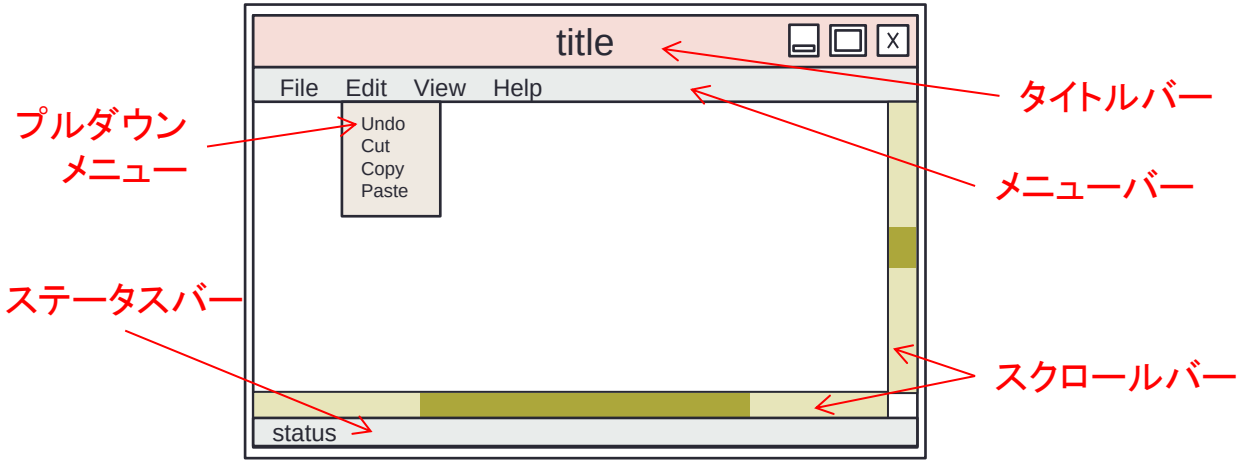


- GUI
  - GUI = Graphical User Interface
  - ポインティングデバイスによる入力
    - マウス，ジョイスティック，トラックボール
    - タッチパッド，タッチパネル
  - コンピュータグラフィックスによる線画や画像の出力

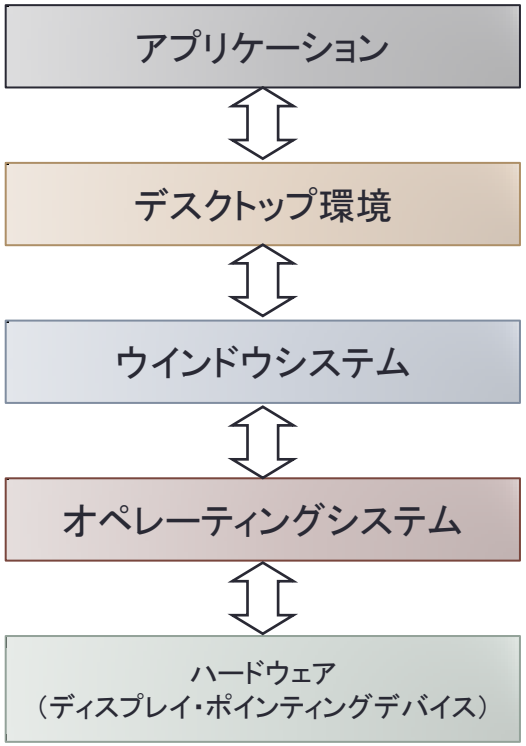


ウインドウシステム

- ウインドウ
  - グラフィカルディスプレイの画面を分割してウインドウを表示
  - あるタスク(プロセス)が利用する固有の領域

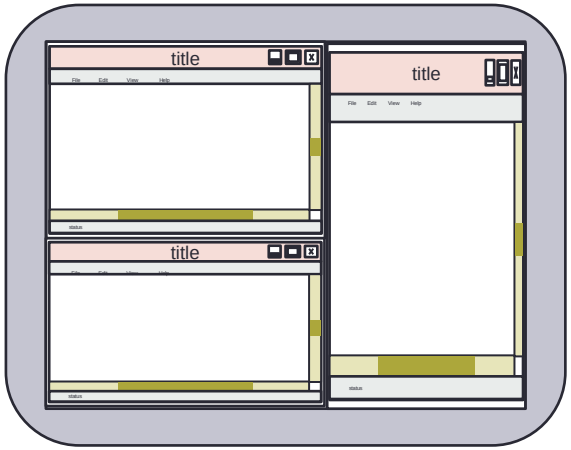


- ウインドウシステム
  - ウインドウを管理するミドルウェア
  - UNIX, Linux: X Window System
  - Mac OS X: Quartz Composer
  - Windows: OSに組み込み

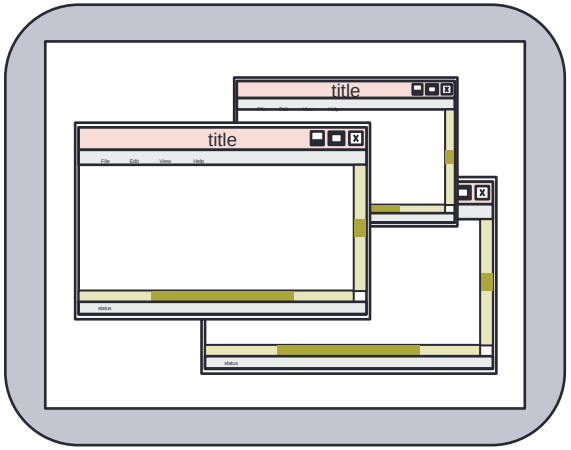


オーバーラッピングウインドウ

- タイリングウインドウ
  - 初期のウインドウシステム
  - ウインドウを重ならないように配置
  - 画面分割に近い
  - すべてのウインドウの情報を同時に見ることができる



- オーバーラッピングウインドウ
  - 現在のウインドウシステムのほとんど
  - ウインドウが重なる
  - 上のウインドウが下のウインドウを隠す
  - 透過するウインドウもある

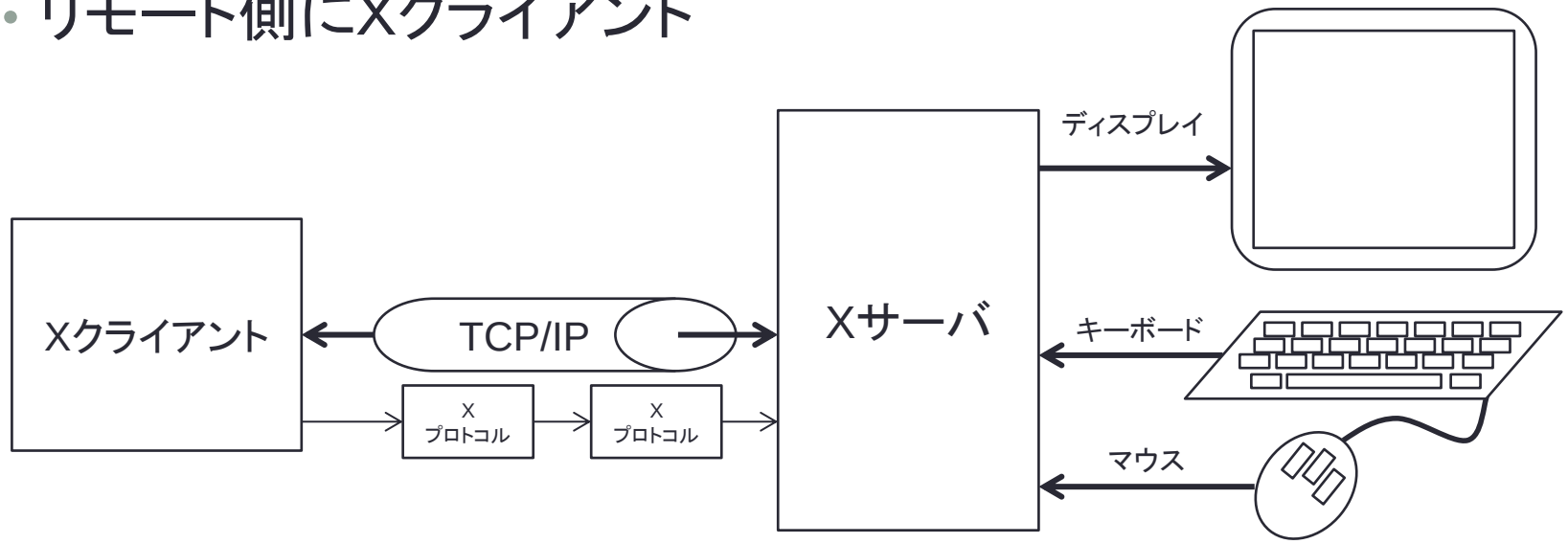


X Window System

- 1984年にMITにおいて開発
  - Project Athenaの一環
  - MITの教育環境上1万台規模ワークステーション上で構築する
  - X10→X11
  - 現在のプロトコルのバージョンが11のために，X11と呼ばれることも多い
- UNIX・Linux用のウインドウシステムとして広く用いられる
  - マルチプラットフォーム
  - Mac OS Xにも初期はバンドル，現在も追加機能として存在
  - Windows上でも動作
- 特徴
  - TCP/IP ネットワークを利用
  - サーバ・クライアントで動作
  - ネットワーク越しにウインドウを表示

サーバ・クライアント

- ユーザ側にXサーバ
- リモート側にXクライアント



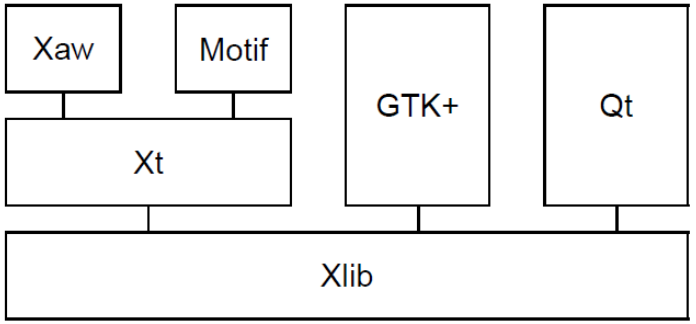
- Xプロトコル
  - XサーバとXクライアント間のプロトコル
  - Xクライアント→Xサーバ
    - 描画の命令をネットワークパケットとして送る
  - Xサーバ→Xクライアント
    - イベントの通知（キーボード入力・マウスイベントなど）

X11の構成要素(1)

- Xサーバ
  - キーボードやマウスなどの入力デバイスを持つ
  - ディスプレイなどの出力デバイスを持つ
  - ウインドウの管理
  - アプリケーションから送られてくるウインドウへの図形や文字などの描画の処理
  - キーボード・マウスからの入力をうけとりイベントへの変換しアプリケーションに渡す

- Xクライアント
  - Xプロトコルを用いてXサーバに接続
  - ウインドウの作成，図形や文字の描画などを行なうアプリケーション
  - Xサーバと同一計算機上にある場合にはUNIX Domain Socketや共有メモリなどを用いて通信
  - 同一計算機上にない場合には，TCP/IPを利用してサーバと通信
  - デフォールトではDISPLAY環境変数によりサーバを指定

X11の構成要素(2)



- Xライブラリ
  - Xサーバと通信するライブラリ
  - Xプロトコルの組み立て
  - サーバからの返答とイベントの分離
  - Xプロトコルそのままの低レベルな命令のみ存在
- Xツールキット
  - Xライブラリ上にボタンやさまざまな部品を作る
  - ウィジェットを作り提供するライブラリ
- Xウインドウマネージャ
  - デスクトップ上のウインドウの配置や移動などの操作をつかさどるクライアントアプリケーション
  - 複数のウインドウマネージャがあり，ユーザの好みで選択
    - ウインドウマネージャによってデスクトップ環境が切り替わる
    - gnome
    - kde

クライアントプログラム例

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
main () {
  Display *d;
  Window w;
  d = XOpenDisplay(NULL);
  w = XCreateSimpleWindow(d, DefaultRootWindow(d), 50, 50,
                          400, 300, 2,
                          BlackPixel(d, 0), WhitePixel(d, 0));
  XMapWindow(d, w);
  XFlush(d);
  getchar();
  XCloseDisplay(d);
}
```

Xサーバに接続

ウインドウ作成

ウインドウ表示

ライブラリ内のデータをサーバに送る

Xサーバとの接続終了

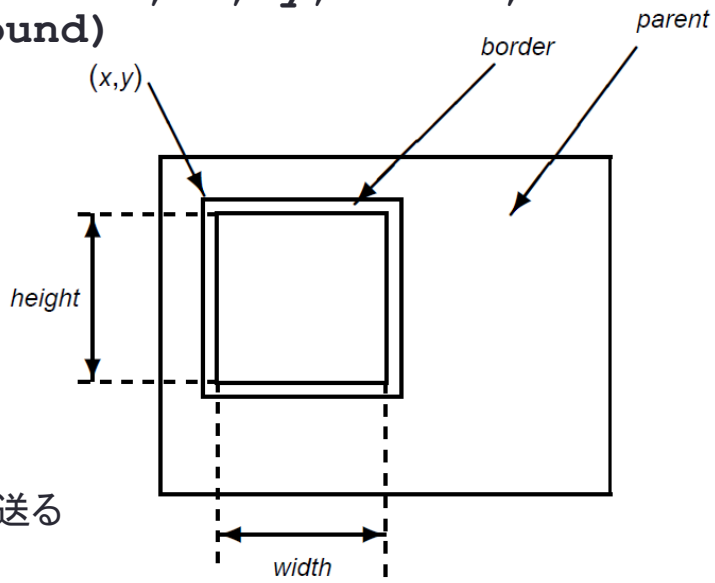
```
% gcc -I/usr/X11R6/include -L/usr/X11R6/lib a.c -lX11
```

Xlib(1)

- **Display *XOpenDisplay("ディスプレイ名")**
  - Xサーバとの通信を開始する.
  - ディスプレイの名前を**NULL**とすると環境変数**DISPLAY**によっ
  - て設定されたXサーバと接続
  - 特定のXサーバとの接続

```
d = XOpenDisplay("host:0.0");
```

- **Window XCreateSimpleWindow(display, parent, x, y, width, height, border width, border, background)**
  - ウインドウの作成
  - **parent**の子ウインドウ
  - 位置や背景色などを指定
  - ウインドウを作っても直ちには表示されない



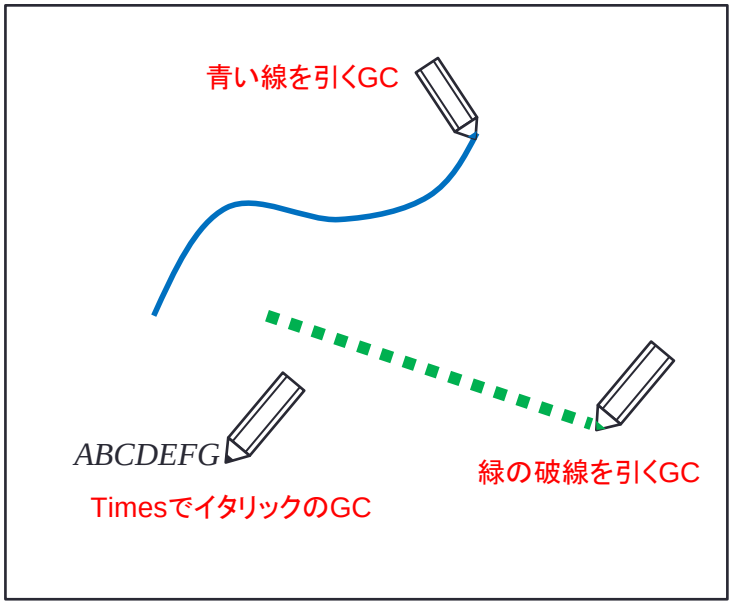
- **XMapWindow(display, window)**
  - **window** を実際に表示する

- **XFlush(display)**
  - クライアントにキャッシュされている命令をサーバに強制的に送る

- **XCloseDisplay(display)**
  - X Server の利用を終了
  - 関連するリソースはすべて解放

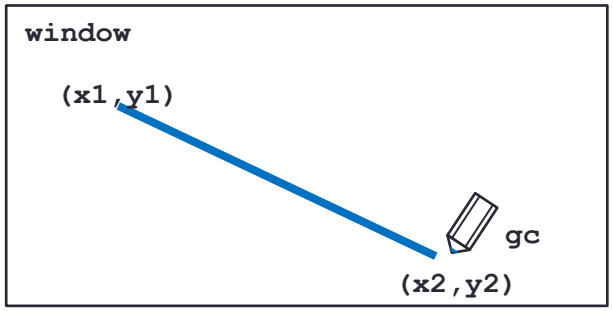
グラフィックスコンテキスト

- 描画するときの色や線の種類，フォントなどの情報をまとめたもの
  - 描画命令ごとに指定すると面倒
  - グラフィックスコンテキストで設定してから描画する
  - 筆やペンにあたる
  - GC
- グラフィックスコンテキストに記憶されるもの
  - 描画色
  - 背景色
  - 線の種類（幅，破線）
  - 線の端点，角の処理
  - 塗りつぶしのスタイル
  - フォント
  - クリッピング情報

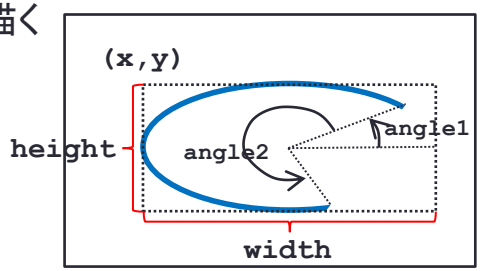


Xlib(2)

- **GC XCreateGC(display, window, mask, values)**
  - **window**に対するグラフィックスコンテキストを新しく作成
  - グラフィックスコンテキスト= 筆
  - 描画対象に対応したグラフィックスコンテキストが必要



- **XSetForeGround(display, gc, pixel)**
  - **pixel**で**gc**の描画色を指定
- **XDrawLine(display, window, gc, x1, y1, x2, y2)**
  - (**x1**, **y1**) から(**x2**, **y2**) まで直線を描画
- **XDrawPoint(display, window, gc, x, y)**
  - **(x, y)**に点を描画
- **XDrawRectangle(display, window, gc, x, y, width, height)**
  - **(x, y)**を左上角とする幅width, 高さheight の四角を描く
- **XDrawArc(display, window, gc, x, y, width, height, angle1, angle2)**
  - **(x, y)**を左上角とする幅**width**, 高さ**height**の四角に接する楕円を描く
  - 楕円の開始角度**angle1**
  - 楕円の角度**angle2**
  - 角度は度数×64で指定



線を描画するプログラム

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
main () {
  Display *d;
  Window w;
  GC gc;
  d = XOpenDisplay(NULL);
  w = XCreateSimpleWindow(d, DefaultRootWindow(d), 50, 50,
                          500, 400, 2,
                          BlackPixel(d, 0), WhitePixel(d, 0));
  XMapWindow(d, w);
  XFlush(d);
  getchar();
  gc = XCreateGC(d, w, 0, 0);
  XSetForeground(d, gc, BlackPixel(d, 0));
  XDrawLine(d, w, gc, 50, 100, 450, 200);
  XFlush(d);
  getchar();
  XCloseDisplay(d);
}
```

GCの作成

描画色の設定

直線を描画

Xlib(3)

- **XFillRectangle(display, window, gc, x, y, width, height)**
  - **XDrawRectangle** と同じだが，中身を塗りつぶす
- **XFillArc(display, window, gc, x, y, width, height, angle1, angle2)**
  - **XDrawArc** と同じだか，中身が塗りつぶす
- **Font XLoadFont(display, name)**
  - 指定された名前のフォントを探す
  - **xslfonts** でX Server のフォントの一覧は見ることができる
  - 拡大縮小可能なフォントの変形などは名前により指定
- **XSetFont(display, gc, font)**
  - フォントをグラフィックスコンテキストに設定
- **XDrawString(display, window, gc, x, y, string, length)**
  - (**x**, **y**) の位置をベースとして文字列string を描画

文字を表示するプログラム

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
main () {
  Display *d;
  Window w;
  GC gc;
  XSetWindowAttributes a;
  Font f;
  d = XOpenDisplay(NULL);
  w = XCreateSimpleWindow(d,DefaultRootWindow(d), 50, 50, 500, 400, 2,
                          BlackPixel(d, 0), WhitePixel(d, 0));
  a.override_redirect = 1;
  XChangeWindowAttributes(d, w, CWOverrideRedirect, &a);
  XMapWindow(d, w);
  XFlush(d);
  f = XLoadFont(d, "a14");
  gc = XCreateGC(d, w, 0, 0);
  XSetForeground(d, gc, BlackPixel(d, 0));
  XSetFont(d, gc, f);
  XDrawString(d, w, gc, 100, 100, "I love SFC.", 11);
  XFlush(d);
  getchar();
  XCloseDisplay(d);
}
```

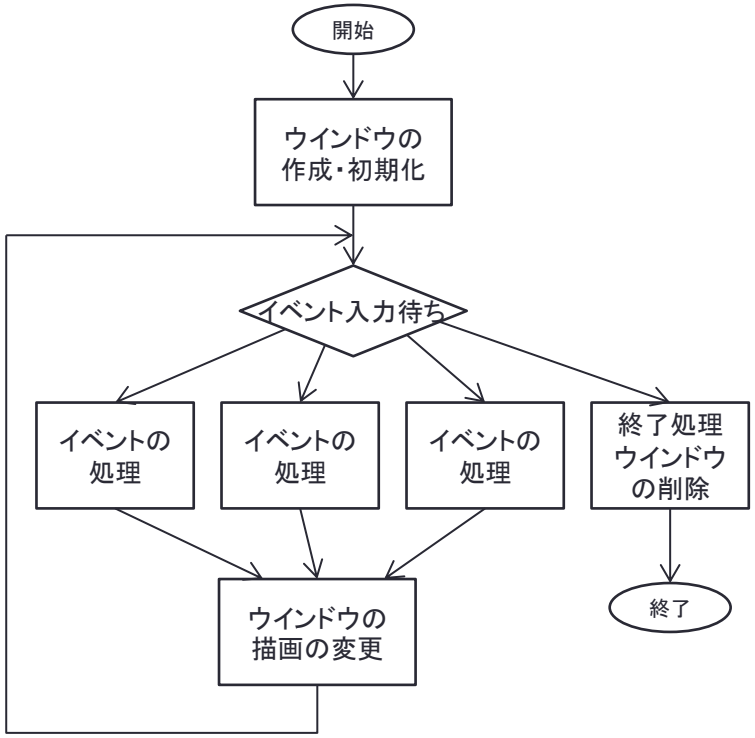
フォントを読み込む

GCにフォントを設定

文字列を描画

イベント処理

- ウインドウシステムの入力はイベントとして与えられる
  - CUIでは文字列が順番に入力される
  - ボタンやメニューなど順不同で操作される
- イベント
  - キーボード関係のイベント
    - **KeyPress**
    - **KeyRelease**
  - マウス関係のイベント
    - **ButtonPressed**
    - **ButtonRelease**
    - **MotionNotify**
  - ウインドウ関係のイベント
    - **Expose**
    - **Map**
    - **Resize**
- イベント処理ループ
  - イベントを待ち，それぞれのイベント毎に処理する
  - イベントは非同期・順不同で発生する
  - CUIのような入力ごとの入力待ちはなく，イベント待ちが一つだけ



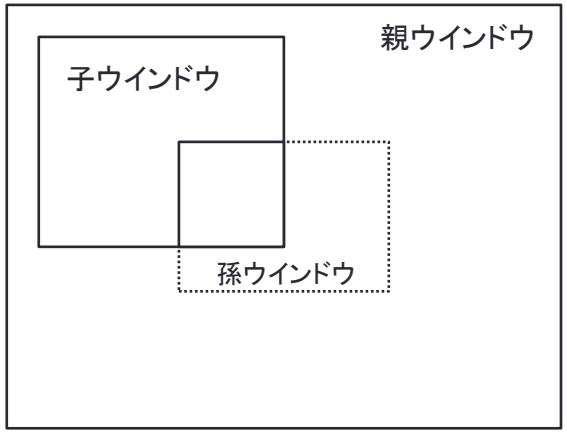
XNextEventによる入力の処理

```
#include<X11/Xlib.h>
#include <X11/Xutil.h>
#include<stdio.h>
main() {
  Display *d;
  Window w;
  XEvent e;
  GC gc;
  int bye;
  d = XOpenDisplay(NULL);
  w = XCreateSimpleWindow(d, DefaultRootWindow(d), 10, 10, 100, 100, 1,
                          BlackPixel(d, 0), WhitePixel(d, 0));
  XSelectInput(d, w, ExposureMask | KeyPressMask);
  XMapWindow(d, w);
  gc = XCreateGC(d, w, 0, 0);
  XSetForeground(d, gc, BlackPixel(d, 0));
  bye = 0;
  while(!bye) {
    XNextEvent(d, &e);
    switch (e.type) {
    case Expose:
      XFillRectangle(d, w, gc, 20, 20, 40, 40);
      break;
    case KeyPress:
      bye = 1;
      break;
    }
  }
  XCloseDisplay(d);
}
```

次のイベントを待つ

イベントに応じた処理

Xサーバのウインドウ

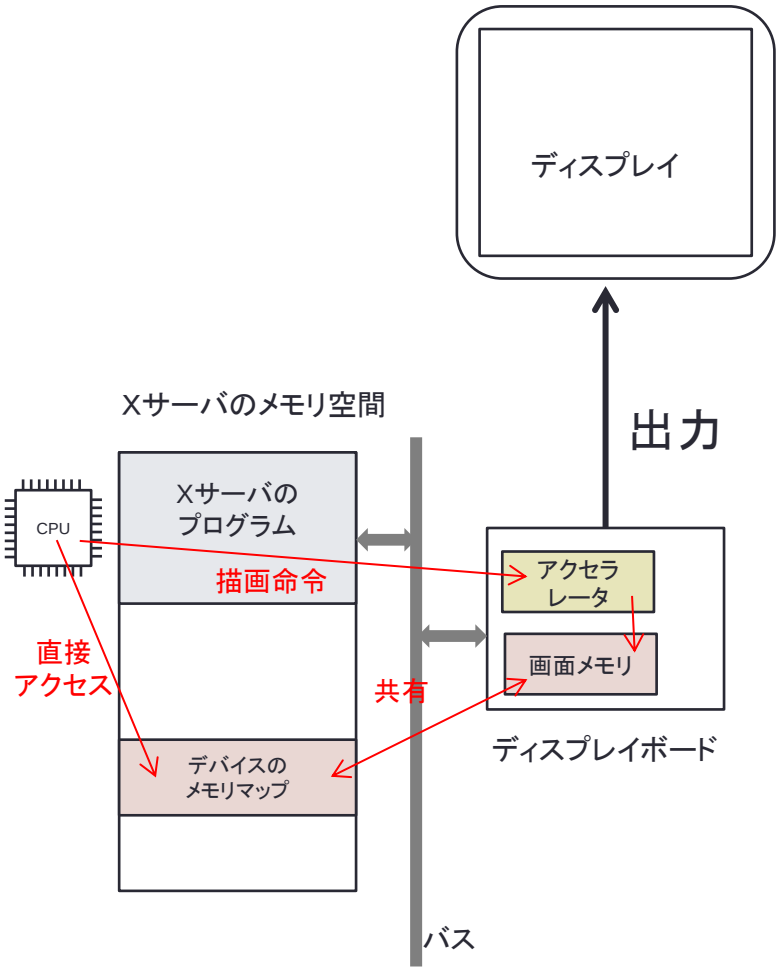


- ウインドウは階層構造
  - 親ウインドウは任意個の子ウインドウを持つ
  - 子ウインドウもまた子ウインドウを持つことが可能
  - 子ウインドウは完全に親ウインドウに含まれる
- ルートウインドウ
  - デスクトップに対応するウインドウ
  - ウインドウ直下の子ウインドウが通常のアプリケーションが作ったウインドウ
  - ウインドウマネージャが装飾を追加することも可能
- ボタンもウインドウ
  - メニューもウインドウ
- Xサーバではウインドウの重なりを調べ，可視部分のみ描画
  - 透過ウインドウの処理は少し大変

描画

- ディスプレイへの描画
  - ハードウェアアクセラレータを利用
  - メモリマップされた値を変更する
  - ディスプレイ上の点を配列の要素としてアクセスできる

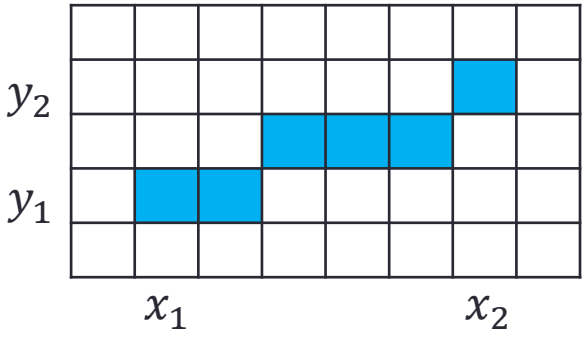
```
int *base = ディスプレイのメモリマップアドレス;
pixel = (R<<16) + (G<<8) + B;
for (y = 0; y < 768; y++) {
  for (x = 0; x < 1024; x++) {
    base[x + y * 1024] = pixel;
  }
}
```



直線の描画

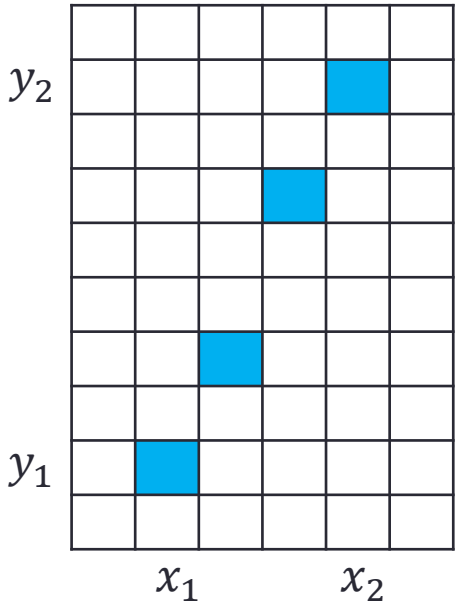
- (x1, y1) から(x2, y2) への直線の描画
- 直線の方程式を利用

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$




```
double a = (double)(y2 - y1)/(x2 - x1);
for (x = x1; x <= x2; x++) {
  y = (int)(a * (x - x1) + y1 + 0.5);
  putPixel(x, y);
}
```

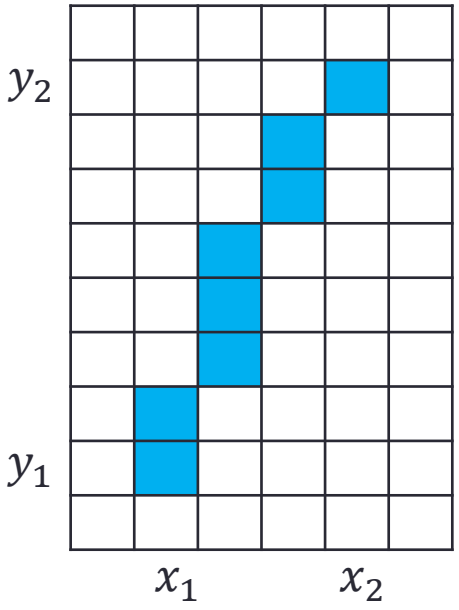
- a > 1 のときにはすき間ができる
- 浮動小数点演算が必要



プレゼンハムのアルゴリズム

- 整数演算だけで直線を描画

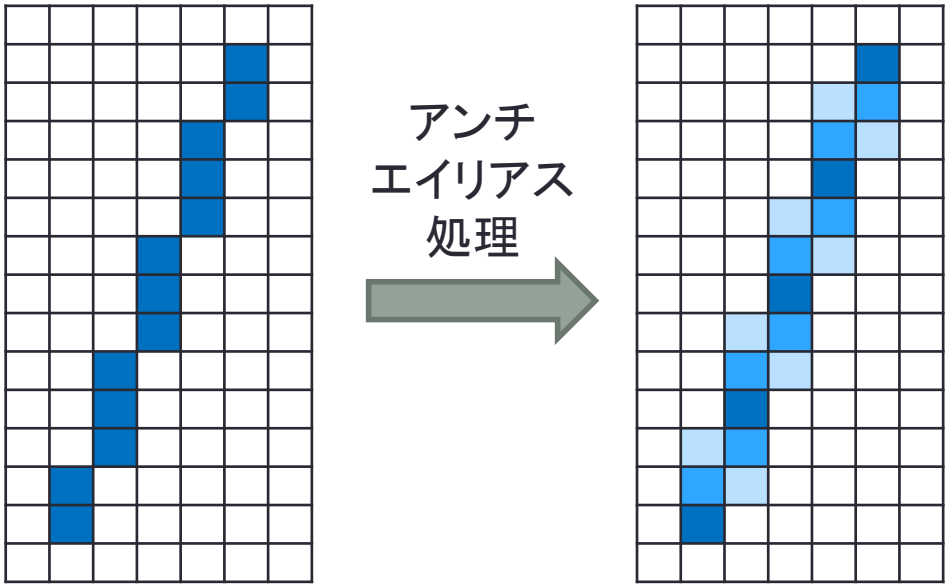
```
int dx, dy, e;
dx = x2 - x1;
dy = y2 - y1;
e = 2 * dy - dx;
y = y1;
for (x = x1; x <= x2; x++) {
  putpixel(x, y);
  if (e >= 0) {
    y++;
    e += 2 * (dy - dx);
  }
  else e += 2 * dy;
}
```



アンチエイリス

- エイリアス
  - 斜めの線にはギザギザ（ジャギー）が発生
  - フォントの描画の場合も同様

- アンチエイリアス
  - 描画色と背景を融合することでジャギーを軽減させる
  - 輪郭がぼやける
  - ビットマップフォントでは行われないことが多い
  - デジタル信号のサンプリングによる折り返しをローパスフィルタで除去することと同等



その他のウインドウシステム

- NeWS
  - Network extensible Window System
  - ウインドウ内の描画命令はすべてDisplay Postscript
  - 画面解像度やディスプレイの種類から独立

- QuickDraw
  - AppleがQuartz以前に用いていた2 次元グラフィックスの描画
  - ライブラリ
  - さまざまな変換などを施して描画が可能
  - 解像度などからは独立してた描画命令
  - 画面上でのカットアンドペーストはQuickDrawの命令列として行なわれる

まとめ

- ウインドウシステム
- X11ウインドウシステム
- 参考書
  - 「Xlib reference manual: for version 11 of the X Window System」, Adrian Nye, 1989.
    - http://www.archive.org/details/xlibretmanver1102nye
  - 「Xlib Reference Manual for Version 11 Vol. 2」, Adrian Nye, 1990.
    - http://www.archive.org/details/xlibrefmanv115ed02