ゲームプログラミング

▶　第６回　文字の描画　◀

Windowsで文字やビットマップを描画するときは、デバイスコンテキストという機能を使います。デバイスコンテキストはさまざまなデバイスがサポートしており、同じAPIで異なるデバイスに描画することができるので、デバイスを意識しないで描画することができます。

デバイスコンテキスト

文字やグラフィックを描画する場合は、描画したいデバイスからデバイスコンテキスト(DC)のハンドルを取得し、それを描画用のAPIに渡すことで行います。

デバイスコンテキストは、ディスプレイやプリンタといった出力デバイスの描画属性に関する情報を管理するものです。デバイスコンテキストが取得できるデバイスなら、どのようなデバイスに対しても同じAPIで描画することができます。

ＧＤＩ(グラフィック・デバイス・インタフェース)

デバイスコンテキストで描画を行う関数群をGDI(グラフィック・デバイス・インタフェース)と呼びます。GDIには文字を描画するTextOut関数やDrawText関数、線を引くLintToEx関数、円を描くEllipse関数、多角形を描くPolygon関数、ビットマップを転送するBitBlt関数、StretchBlt関数など高機能な関数がたくさんありまず。

デバイスコンテキストの取得と解放

クライアント領域のデバイスコンテキストのハンドルはGetDC関数で取得します。この関数の引数はウィンドウのハンドルです。関数が正常に終了するとデバイスコンテキストのハンドルが返ります。このハンドルをGDI関数に渡すことで描画を行うことができます。

取得したデバイスコンテキストは、必要がなくなったらReleaseDC関数で解放します。この関数の１つ目の引数はウィンドウのハンドル、２つ目の引数は解放するデバイスコンテキストのハンドルです。

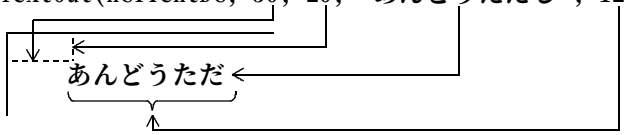
```
デバイスコンテキストの取得：HDC   hClientDC = GetDC(hWnd); // hWndはウィンドウのハンドル
デバイスコンテキストの解放：ReleaseDC(hWnd, hClientDC);
```

TextOut関数，DrawText関数

文字の描画は、TextOut関数とより高機能なDrawText関数で行うことができます。

TextOut関数は、単純に文字を描画する関数で、改行(\n)を処理することができません。この関数の１つ目の引数は描画対象のデバイスコンテキストのハンドル、２つ目の引数は描画を開始するｘ座標、３つ目の引数は描画を開始するｙ座標、４つ目の引数は描画文字列、５つ目の引数が描画する文字数です。char型の文字は半角を１文字、全角を２文字として数えます。lstrlen関数を使えば、文字列から文字数を求めることができます。

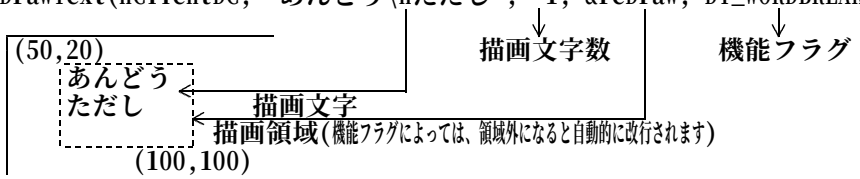
```
TextOut(hClientDC, 50, 20, "あんどうただし", 12);
```



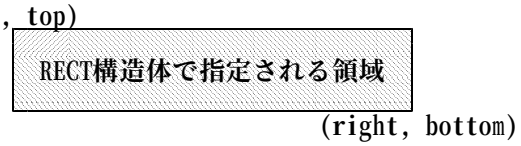
DrawText関数は、改行やタブといった制御文字も処理でき、さらに描画領域の指定、日本語禁則処理、自動改行、右揃え、左揃え、センタリングなどさまざまな機能をサポートします。高機能なぶん、TextOut関数より描画速度は低下します。

この関数の１つ目の引数は描画対象のデバイスコンテキストのハンドル、２つ目の引数は描画文字列、３つ目の引数は文字列の長さ、４つ目の引数は描画領域、５つ目の引数は描画フラグです。文字列の長さは-1を指定するとすべての文字が描画されます。描画フラグは禁則処理やセンタリングなどの機能を指定します。

```
RECT    rcDraw = {50, 20, 100, 100};      // 描画領域
DrawText(hClientDC, "あんどう\nただし", -1, &rcDraw, DT_WORDBREAK);
```



※RECT構造体
　領域を指定するときによく使うのがRECT構造体です。この構造体は、left、top、right、bottomの４つのメンバで構成されています。以下のように、leftとtopメンバで左上の座標、rightとbottomメンバで右下の座標となり、これを対角の頂点とする長方形の領域を指定します。



文字の描画

　文字は以下の手順で描画します。

(１)デバイスコンテキストのハンドルを取得　　GetDC関数
(２)文字描画　　TextOut関数, DrawText関数
(３)デバイスコンテキストの解放　　ReleaseDC関数

```
HDC   hClientDC = GetDC(hWnd);
if(NULL != hClientDC) {
    LPCTSTR   lpszText = "あんどうただし";  // 描画文字列
    TextOut(hClientDC, 0, 0, lpszText, lstrlen(lpszText));
    ReleaseDC(hWnd, hClientDC);
}
```

文字色と背景の変更

　文字色の変更はSetTextColor関数、背景色の変更はSetBkColor関数、背景モードの変更はSetBkMode関数で行います。

```
SetTextColor(hClientDC, RGB(128, 128, 128));      // 文字色を灰色に設定
SetBkColor(hClientDC, RGB(0, 0, 255));            // 背景色を青に設定
SetBkMode(hClientDC, TRANSPARENT));               // 背景モードは透過
```

※RGB()は色の指定に使うマクロで、左から赤、緑、青の輝度を０～２５５の値で指定します。
※背景モードが透過モードのときは、背景色を設定しても透明になります。

フォントの変更

　デバイスコンテキストによる文字の描画は、デフォルトフォントで行われます。フォントの形やサイズなどを変更して描画したい場合は、以下の手順で新しいフォントを生成し、選択する必要があります。

(１)フォントの定義－LOGFONT構造体－
　フォントの大きさや形状などの属性をLOGFONT構造体に定義します。大きさ形状のほか、イタリック体にしたり、アンダーライン、打ち消し線を付けることもできます。

(２)フォントの生成－CreateFontIndirect関数, CreateFont関数－
　(１)で定義した構造体をCreateFontIndirect関数に渡すと、新しいフォントが生成され、そのフォントのハンドルが返されます。(１)と(２)を行うCreateFont関数もあります。

(３)フォントの選択－SelectObject関数－
　(２)で生成したフォントは、SelectObject関数でデバイスコンテキストに選択させることができます。この関数の１つ目の引数はデバイスコンテキストのハンドル、２つ目の引数は新たに選択するオブジェクト(フォント、ペン、ブラシ、ビットマップなど)のハンドルです。関数が成功すると、選択前のオブジェクトのハンドルが返されます。このハンドルがデフォルトオブジェクトのものである場合は、必ず保存しておきます。

(４)文字描画－TextOut関数, DrawText関数－
　生成したフォントが選択された状態で文字を描画すると、そのフォントで描画されます。いろいろなフォントを生成しておき、必要に応じて変更しながら描画することもできます。

(５)デフォルトフォントに戻す－SelectObject関数－
　Windowsは一度に所有できるハンドル数に制限があるため、生成したフォントが不用になったら解放します。このとき、デバイスコンテキストは必ずなんらかのフォントが選択されていないといけないので、生成したフォントを選択したまま解放を行うのは非常に危険です。解放前に、(３)で保存しておいたデフォルトフォントや別のフォントを選択してから解放します。

(６)生成したフォントの解放－DeleteObject関数－
　デフォルトフォント(または別のフォント)を選択したら、生成したフォントをDeleteObject関数で解放します。

```
// フォント定義
LOGFONT   lfFont;
ZeroMemory(&lfFont, sizeof(lfFont));
lfFont.lfHeight         = 64;                   // 高さ
lfFont.lfWidth          =  0;                   // 幅
lfFont.lfEscapement     =  0;                   // 行の角度
lfFont.lfOrientation    =  0;                   // ベースラインの角度
lfFont.lfWeight         =  0;                   // 太さ
lfFont.lfItalic         = FALSE;                // 斜体
lfFont.lfUnderline      = FALSE;                // アンダーライン
lfFont.lfStrikeOut      = FALSE;                // 打ち消し線
lfFont.lfCharSet        = SHIFTJIS_CHARSET;     // キャラクターセット
                                                // (英字フォントならDEFAULT_CHARSETにする)
lfFont.lfOutPrecision   = OUT_DEFAULT_PRECIS;   // 出力精度
lfFont.lfClipPrecision  = CLIP_DEFAULT_PRECIS;  // クリッピング精度
lfFont.lfQuality        = DEFAULT_QUALITY;      // 出力品質
lfFont.lfPitchAndFamily = DEFAULT_PITCH;        // ピッチとファミリ
lstrcpy(lfFont.lfFaceName, "ＤＦＰまるもじ体");  // 文字フォント名(コントロールパネルのフォントと
                                                // 一字一句同じに設定しないと正しく選択されない)
// フォント生成
HFONT   hFont, hDefFont;
hFont = CreateFontIndirect(&lfFont);
if(NULL != hFont)
    // フォント選択
    hDefFont = (HFONT)SelectObject(hClientDC, hFont);

// 文字描画
RECT   rcText = {100, 100, 320, 400};
DrawText(hClientDC, "あんどう\nただし", -1, &rcText, DT_WORDBREAK);

// フォント解放
if(NULL != hDefFont) {
    SelectObject(hClientDC, hDefFont);  // デフォルトフォントに戻す
    DeleteObject(hFont);                // フォント解放
}
```

フォントファイルの読み込み
　コントロールパネルの「フォント」に登録されていないフォントでも、フォントファイルがあれば、AddFontResource関数で読み込んでシステムに登録することができます。登録したフォントは「フォントの変更」と同じ手順で描画することができます。システムに登録したフォントは、不用になったらRemoveFontResource関数で削除します。これらの関数の引数は、フォントのファイル名です。
　フォントファイル読み込み：AddFontResource("D:\\Font\\HGRPP1.TCC");
　フォントファイル読み込み解除：RemoveFontResource("D:\\Font\\HGRPP1.TCC");

縁取り
　文字のまわりが縁取られているような文字を描画したい場合は、背景モードを「透過」にしてから、縁取り部分を中心から上下左右にずらして描画します。そのあと中心の文字を描画することで縁取り文字の完成です。

```
LPCTSTR     DRAW_TEXT = "アンドウタダシ";     // 描画文字
const int   DRAW_X = 200, DRAW_Y = 100;      // 描画座標
SetBkMode(hClientDC, TRANSPARENT);           // 背景モードを「透過」に設定

// 縁取り描画
SetTextColor(hClientDC, RGB(255, 255, 255));
TextOut(hClientDC, DRAW_X - 1, DRAW_Y,     DRAW_TEXT, lstrlen(DRAW_TEXT));  // 左にずらして描画
TextOut(hClientDC, DRAW_X + 1, DRAW_Y,     DRAW_TEXT, lstrlen(DRAW_TEXT));  // 右にずらして描画
TextOut(hClientDC, DRAW_X,     DRAW_Y - 1, DRAW_TEXT, lstrlen(DRAW_TEXT));  // 上にずらして描画
TextOut(hClientDC, DRAW_X,     DRAW_Y + 1, DRAW_TEXT, lstrlen(DRAW_TEXT));  // 下にずらして描画

// 中心文字描画
SetTextColor(hClientDC, RGB(0, 255, 0));
TextOut(hClientDC, DRAW_X, DRAW_Y, DRAW_TEXT, lstrlen(DRAW_TEXT));
```

○ 課題 ○

ウィンドウのクライアント領域にTextOut関数とDrawText関数で文字を描画してみよう。描画できたら、文字色、背景色、背景モード、フォントの形・大きさを変更してみましょう。また、縁取り文字も描画してみましょう。