ゲームプログラミング

▶　第７回　ビットマップの描画　◀

ビットマップファイルを描画するには、いくつかのAPIを組み合わせて行う必要があり、多少複雑な手順をとります。

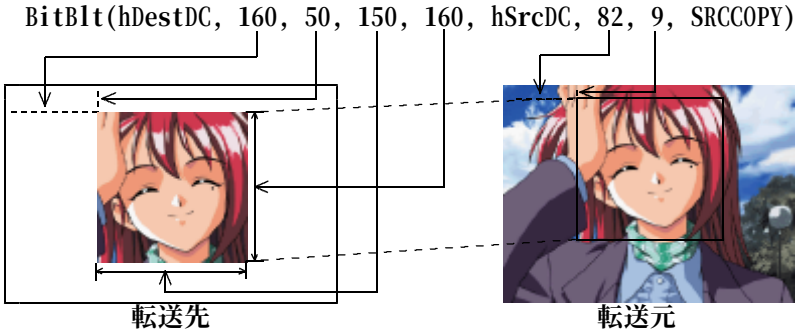
BitBlt関数とStretchBlt関数

ビットマップを転送するAPIに、BitBlt関数とStretchBlt関数があります。これらの関数は、ビットマップをデバイスコンテキスト経由で転送するので、デバイスコンテキストをサポートしていれば、どのようなデバイスでも転送することができます。

BitBlt関数

BitBlt関数は、転送元デバイスコンテキストから転送先デバイスコンテキストに、ビットマップデータをブロック転送します。引数は先頭から以下のようになります。

```
HDC   hdcDest;   /* 転送先デバイスコンテキストのハンドル */
int   nXDest;    /* 転送先ｘ座標                         */
int   nYDest;    /* 転送先ｙ座標                         */
int   nWidth;    /* 転送するビットマップの幅             */
int   nHeight;   /* 転送するビットマップの高さ           */
HDC   hdcSrc;    /* 転送元デバイスコンテキストのハンドル */
int   nXSrc;     /* 転送元ｘ座標                         */
int   nYSrc;     /* 転送元ｙ座標                         */
DWORD dwRop;     /* ラスタオペレーションコード(ＲＯＰ)   */
```



最後の引数の「ラスタオペレーションコード」は、転送元と転送先のビットマップデータをどのように結合するかという指定です。通常はSRCCOPYを使用して単純に転送元から転送先にコピーするだけですが、画像を反転したり、特定の色で塗りつぶしたりといった効果を付加して転送することができます。また、独自のコードを作成することもできます。

ラスタオペレーションコード	演　　算	意　　味
SRCCOPY	転送先 = 転送元	転送元を上書き
NOTSRCCOPY	転送先 = ~転送元	転送元のビット列を反転して上書き
SRCPAINT	転送先 = 転送先 | 転送元	転送先と転送元のビット列のＯＲを取る
SRCAND	転送先 = 転送先 & 転送元	転送先と転送元のビット列のＡＮＤを取る
SRCINVERT	転送先 = 転送先 ^ 転送元	転送先と転送元のビット列のＸＯＲを取る
SRCERASE	転送先 = ~転送先 & 転送元	転送元のビット列を転送先のビット列から消去

▲代表的なラスタオペレーションコード

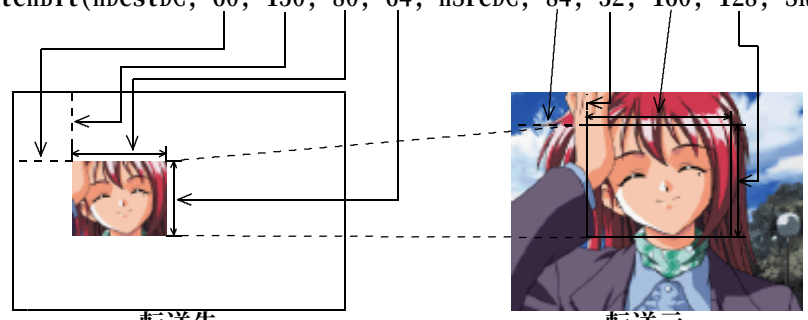
StretchBlt関数

StretchBlt関数は、転送元領域から転送先領域へビットマップデータをブロック転送します。転送元と転送先で領域の大きさが異なる場合、転送先領域の大きさに合わせてビットマップを拡大縮小して転送します。引数は先頭から以下のようになります。

```
HDC   hdcDest;      /* 転送先デバイスコンテキストのハンドル */
int   nXOriginDest; /* 転送先ｘ座標                         */
int   nYOriginDest; /* 転送先ｙ座標                         */
```

```
int   nWidthDest;   /* 転送先領域の幅                          */
int   nHeightDest;  /* 転送先領域の高さ                        */
HDC   hdcSrc;       /* 転送元のデバイスコンテキストのハンドル  */
int   nXOriginSrc;  /* 転送元ｘ座標                            */
int   nYOriginSrc;  /* 転送元ｙ座標                            */
int   nWidthSrc;    /* 転送元領域の幅                          */
int   nHeightSrc;   /* 転送元領域の高さ                        */
DWORD dwRop;        /* ラスタオペレーションコード              */
```

```
StretchBlt(hDestDC, 60, 150, 80, 64, hSrcDC, 84, 32, 160, 128, SRCCOPY)
```



転送先　　転送元

ビットマップファイルの読み込み

ファイルやリソースにあるビットマップは、メモリに読み込まなければ転送できません。ビットマップを読み込む関数に、LoadImage関数があります。この関数は、ビットマップのほかにカーソル、アイコンといったイメージを読み込み、そのハンドルを得ることができます。同じような動作をするAPIに、LoadBitmap関数、LoadCursor関数、LoadIcon関数がありますが、これらのAPIはリソースからしか読み込むことができません。LoadImage関数の引数は、以下のようになります。

```
LoadImage(アプリケーションのインスタンスハンドル，ロードするイメージ名，イメージのタイプ，
          イメージの幅，イメージの高さ，ロードフラグ)
```

１つ目の引数は、アプリケーションのインスタンスハンドルです。ファイルから読み込む場合はNULLを指定します。リソースから読み込むときは、アプリケーションのインスタンスハンドルを指定します。GetModuleHandle(NULL)で代用することもできます。

２つ目の引数は、読み込むファイル名またはリソース名です。ファイルが格納されているパス名やリソース識別子を記述します。

３つ目の引数は、読み込むイメージのタイプです。ビットマップは「IMAGE_BITMAP」、カーソルは「IMAGE_CURSOR」、アイコンは「IMAGE_ICON」を指定します。

４つ目と５つ目の引数は、イメージの幅と高さです。ここで指定した大きさだけ読み込まれます。０を指定すると、イメージ全体が読み込まれます。

６つ目の引数は、フラグです。ビットマップをファイルから読み込むときは「LR_LOADFROMFILE」と「LR_CREATEDIBSECTION」の２つを指定します。

戻り値は、イメージの読み込みに成功するとそのハンドル、それ以外の場合はNULLが返されます。

```
// ビットマップファイル読み込み
HBITMAP   hBMP = (HBITMAP)LoadImage(NULL, "HIKARI.BMP", IMAGE_BITMAP,
                                    0, 0, LR_LOADFROMFILE | LR_CREATEDIBSECTION);
if(NULL == hBMP) {
    // 読み込み失敗(エラー処理)
}
```

読み込んだビットマップファイルは、不用になったらDeleteObject関数で解放します。

```
DeleteObject(hBMP);    // 読み込まれたビットマップの解放
```

デバイスコンテキストの取得

BitBlt関数やStretchBlt関数でビットマップを転送するには、転送先と転送元のデバイスコンテキストが必要になります。ビットマップファイルを読み込んでウィンドウのクライアント領域に転送する場合は、ビットマップファイルとクライアント領域のデバイスコンテキストが必要になります。

クライアント領域のデバイスコンテキストはGetDC関数で取得できますが、ビットマップファイルから直接デバイスコンテキストを取得することはできません。そこで、メモリデバイスコンテキストという特殊なデバイスコンテキストをビットマップ転送のサポート役として利用します。

メモリデバイスコンテキストは、メモリに設けられた擬似的な画面のデバイスコンテキストのことです。これは、CreateCompatibleDC関数で生成することができます。

すべてのデバイスコンテキストは、あらゆる描画のために、デフォルトのペン(線を引く)、フォント(文字を描画)、ブラシ(塗りつぶし)、リージョン(描画領域の指定)といったものの他に、描画データを保持するためのビットマップを持っています。このビットマップは、メモリに読み込まれた任意のビットマップと交換することができます。

LoadImage関数でビットマップファイルをメモリに読み込み、これをメモリデバイスコンテキストに選択します。こうすることにより、メモリデバイスコンテキストをビットマップファイルのデバイスコンテキストのように利用することができます。

```
// メモリデバイスコンテキスト生成
HDC    hMemDC = CreateCompatibleDC(NULL);
if(hMemDC == NULL) {
    // メモリデバイスコンテキスト生成失敗(エラー処理)
}

// 読み込んだビットマップを選択
HBITMAP    hOldBMP = (HBITMAP)SelectObject(hMemDC, hBMP);
if(NULL == hOldBMP) {
    // 選択失敗(エラー処理)
}
```

メモリデバイスコンテキストは､不用になったら最初に保持していたデフォルトビットマップに戻し､DeleteDC関数で解放します。

```
// メモリデバイスコンテキスト解放
SelectObject(hMemDC, hOldBMP);      // デフォルトビットマップに戻す
DeleteDC(hMemDC);                   // メモリデバイスコンテキスト解放
```

ビットマップ情報とクライアント領域の大きさを取得

ビットマップの情報やクライアント領域の大きさを取得したい場合があります。ビットマップの情報はGetObject関数､クライアント領域の大きさはGetClientRect関数でそれぞれ取得することができます。

```
// ビットマップ情報取得
BITMAP    bmBMPInfo;
GetObject(hBMP, sizeof(bmBMPInfo), &bmBMPInfo);

// クライアント領域のサイズ取得
RECT      rcClient;
GetClientRect(hWnd, &rcClient);     // hWndはウィンドウのハンドル
```

ビットマップファイルの描画手順

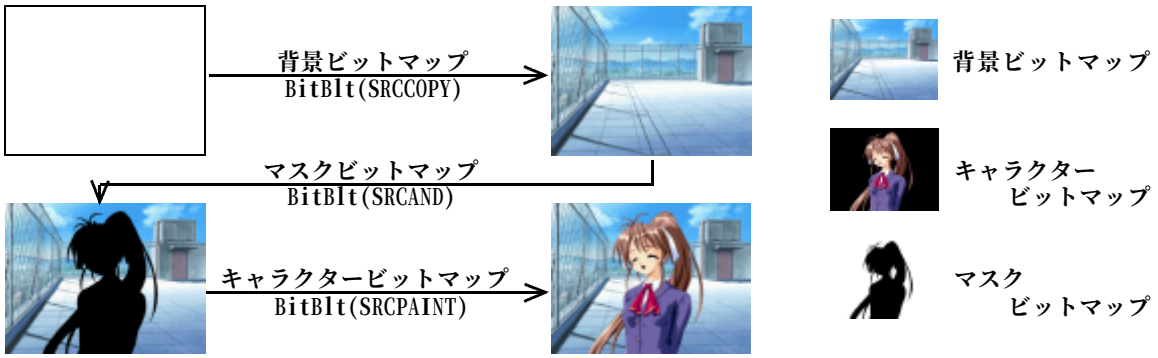
(1)ビットマップの読み込み	LoadImage関数
(2)メモリデバイスコンテキストの生成	CreateCompatibleDC関数
(3)読み込んだビットマップの選択	SelectObject関数
(4)ビットマップ情報の取得	GetObject関数
(5)転送先デバイスコンテキストの取得	GetDC関数
(6)転送	BitBlt関数, StretchBlt関数
(7)転送先デバイスコンテキストの解放	ReleaseDC関数
(8)デフォルトビットマップに戻す	SelectObject関数
(9)メモリデバイスコンテキストの解放	DeleteDC関数
(10)読み込んだビットマップの解放	DeleteObject関数

背景とキャラクターの重ね合わせ

2つのビットマップを重ねて表示しようとした場合、背景ビットマップを描画し、その上に同じ描画処理によりキャラクタービットマップを描画すると、キャラクタービットマップがそのまま描画されることになります。正しく重ねて描画したい場合は、描画したくないピクセルを透明色として設定する必要があります。

透明色の設定を行うためには、透明色のピクセルが白、それ以外の色の部分が黒となるようなマスクビットマップの作成が必要となります。

マスクビットマップを使って重ね合わせるには、まず背景ビットマップを通常のSRCCOPYで描画します。次に、マスクビットマップをSRCANDで描画します。こうすると、マスクビットマップの黒の部分が背景からくりぬかれます。最後にキャラクタービットマップをSRCPAINTで転送して重ね合わせます。



○ 課題 ○

(１)ビットマップファイルを読み込み、指定したデバイス(コンテキスト)に転送するLoadBMP関数を完成させ、ビットマップを描画してみましょう。また、StretchBlt関数で拡大縮小転送も実験してみましょう。

LoadBMP関数は、２つ目の引数(lpszBMPName)で指定されたビットマップファイルを読み込み、ビットマップ全体を１つ目の引数(hDC)で指定されたデバイスコンテキストの位置(0, 0)へBitBlt関数で転送します。３つ目の引数(dwROP)はラスタオペレーションコードです。関数が成功するとtrue、それ以外はfalseを返します。

```
/*****************************************************************************************************/
/*                                          ビットマップ読み込み                                     */
/*****************************************************************************************************/
bool LoadBMP(HDC hDC, LPCTSTR lpszBMPName, DWORD dwROP)
{
    if(NULL == hDC)
        return false;

    // ビットマップ読み込み

    // メモリデバイスコンテキスト生成

    // 読み込んだビットマップを選択

    // ビットマップ情報取得

    // 転送

    // ビットマップオブジェクト解放

    return true;
}
```

―使用例―

```
    HDC   hClientDC = GetDC(hWnd);
    if(NULL != hClientDC) {
        LoadBMP(hClientDC, "D:\\BMP\\HIKARI.BMP", SRCCOPY);
        ReleaseDC(hWnd, hClientDC);
    }
```

(２)以下のサンプル画像「背景」「キャラクター」「マスクパターン」を使い、背景にキャラクターを重ねて描画しましょう。


背景.BMP


キャラクター.BMP


マスクパターン.BMP