オブジェクト指向と
ゲームプログラミング

▶ コンポーネント編 ― 第7回 パス ◀

パス

パスは、輪郭または塗りつぶされたひとつまたは複数の図形および直線、曲線の集まりのことです。パスは、リージョンと違ってハンドルで識別されるオブジェクトではなく、描画関数を実行することによりデバイスコンテキストにひとつだけ作成されます。パスを使ってクリッピング領域を設定することもでき、また、リージョンに変換することもできます。

パスとリージョンはクリッピング領域を扱うという点で非常に似ていますが、パスは、その名のとおり、図形を一筆書きの集まりとして扱うのに対し、リージョンは、図形を多角形の集まり、つまり領域として扱うという点で異なります。

パスの作成

パスを扱うAPIには以下のものがあります。

BeginPath	パスをオープンします
CloseFigure	パス内の開かれている図形を閉じます
EndPath	パスをクローズし、デバイスコンテキストに選択します
FillPath	パスの内部を塗りつぶします
FlattenPath	パスの曲線を座標変換し、直線の集合に変換します
PathToRegion	パスからリージョンを作成します
SelectClipPath	パスを利用してクリッピング領域を変更します
SetMiterLimit	マイター接合の制限値さを設定します
StrokeAndFillPath	パスの輪郭を描画し、内部を塗りつぶします
StrokePath	パスの輪郭を描画します
WidenPath	パスを塗る領域として再定義します

パスを作成するには、BeginPath関数とEndPath関数の間に、何回かの描画を行なえば、それがパスとして記録されます。BeginPath関数とEndPath関数で囲まれた領域を、パスブラケットと呼びます。パスブラケット内での描画関数はパスに変換され、実際には描画されません。パスブラケット内ではすべての描画関数が使用できるわけではなく、有効な関数は以下のものに制限されます。

AngleArc	Arc	ArcTo	Chor	CloseFigure
Ellipse	ExtTextOut	LineTo	MoveToEx	Pie
PolyBezier	PolyBezierTo	PolyDraw	Polygon	Polyline
PolylineTo	PolyPolygon	PolyPolyline	Rectangle	RoundRect
TextOut				

これらの関数を用いて描画された図形がパスとなります。なお、パスはデバイスコンテキストに作成されますが、直接アクセスすることも、ハンドルを取得することもできません。

パスは、FillPath関数、PathToRegion関数、StrokePath関数、StrokeAndFillPath関数を実行すると破棄されます。つまり、パスは一度描画すると破棄されてしまうのです。これに対し、リージョンはハンドルを閉じるまで何度でも使用することができます。パスを保存したい場合は、PathToRegion関数でリージョンに変換することにより行います。

課　　題

ヘルプやMSDNライブラリでパスを扱うAPIについて調べながら、パスの効果を確認しましょう。

(１)文字の沿ったパスを作成し、描画してみましょう。以下のようにCTestScene::ActiveProc関数を変更し、TextOut関数による描画がパスに変更されることを確認しましょう。

```
int CTestScene::ActiveProc()
{
    // ここに、機能テストのメイン処理を記述します
    // 内部処理

    // 描画処理
    if(FPSTimer().IsSkip() == false) {
        // ここに、機能テストの描画処理を記述します
        // 背景転送
        ::BitBlt(m_pBackBuf->GetDC(), 0, 0, m_pBackBuf->GetWidth(), m_pBackBuf->GetHeight(),
                 m_pBG->GetDC(), 0, 0, SRCCOPY);

        // 文字に沿ったパスを作成
        // パス開始
        ::BeginPath(m_pBackBuf->GetDC());

        // フォント作成
        HFONT   hFont    = ::CreateFont(96, 0, 0, 0, FW_BOLD, FALSE, FALSE, FALSE, SHIFTJIS_CHARSET,
                                        OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, 0, NULL);
        HFONT   hOldFont = (HFONT)::SelectObject(m_pBackBuf->GetDC(), hFont);

        // パスに文字列描画
        LPCTSTR   PathStr = "あんどうただし";
        ::SetBkMode(m_pBackBuf->GetDC(), TRANSPARENT);
        ::TextOut(m_pBackBuf->GetDC(), 20, 100, PathStr, ::lstrlen(PathStr));

        // フォント解放
        ::SelectObject(m_pBackBuf->GetDC(), hOldFont);
        ::DeleteObject(hFont);

        // パス終了
        ::EndPath(m_pBackBuf->GetDC());

        // パスの輪郭を描画
        ::StrokePath(m_pBackBuf->GetDC());

        FPSTimer().DrawFPS(m_pBackBuf->GetDC());

        // バックバッファからディスプレイへ転送
        HDC   hDestDC = ::GetDC(GameApp().GetHWnd());
        ::BitBlt(hDestDC, 0, 0, m_pBackBuf->GetWidth(), m_pBackBuf->GetHeight(),
                 m_pBackBuf->GetDC(), 0, 0, SRCCOPY);
        ::ReleaseDC(GameApp().GetHWnd(), hDestDC);
    }

    return 0;
}
```

(２)「パスの輪郭を描画」の部分を以下のプログラムに変更し、動作を確認しましょう。

```
    // パスを塗りつぶして描画
    HBRUSH   hBrush    = ::CreateHatchBrush(HS_DIAGCROSS, RGB(0, 0, 255));
    HBRUSH   hOldBrush = (HBRUSH)::SelectObject(m_pBackBuf->GetDC(), hBrush);

    ::FillPath(m_pBackBuf->GetDC());

    ::SelectObject(m_pBackBuf->GetDC(), hOldBrush );
    ::DeleteObject(hBrush);
```

(３)「FillPath()」を以下のプログラムに変更しましょう。

```
    ::StrokeAndFillPath(m_pBackBuf->GetDC());
```

(４)パスをクリッピング領域として設定し、キャラクターを転送してみましょう。以下のようにCTestScene::ActiveProc関数を変更しましょう。

```
int CTestScene::ActiveProc()
{
    // ここに、機能テストのメイン処理を記述します
    // 内部処理

    // 描画処理
    if(FPSTimer().IsSkip() == false) {
        // ここに、機能テストの描画処理を記述します
        // 背景転送
        ::BitBlt(m_pBackBuf->GetDC(), 0, 0, m_pBackBuf->GetWidth(), m_pBackBuf->GetHeight(),
                 m_pBG->GetDC(), 0, 0, SRCCOPY);

        // 文字に沿ったパスを作成
        // パス開始
        ::BeginPath(m_pBackBuf->GetDC());

        // フォント作成
        HFONT   hFont    = ::CreateFont(96, 0, 0, 0, FW_BOLD, FALSE, FALSE, FALSE, SHIFTJIS_CHARSET,
                                        OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, 0, NULL);
        HFONT   hOldFont = (HFONT)::SelectObject(m_pBackBuf->GetDC(), hFont);

        // パスに文字列描画
        LPCTSTR   PathStr = "あんどうただし";
        ::SetBkMode(m_pBackBuf->GetDC(), TRANSPARENT);
        ::TextOut(m_pBackBuf->GetDC(), 20, 100, PathStr, ::lstrlen(PathStr));

        // フォント解放
        ::SelectObject(m_pBackBuf->GetDC(), hOldFont);
        ::DeleteObject(hFont);

        // パス終了
        ::EndPath(m_pBackBuf->GetDC());

        // パスをクリッピング領域として設定
        ::SelectClipPath(m_pBackBuf->GetDC(), RGN_COPY);

        // キャラクター転送
        ::BitBlt(m_pBackBuf->GetDC(), 0, 0, m_pBackBuf->GetWidth(), m_pBackBuf->GetHeight(),
                 m_pChara  ->GetDC(), 0, 0, SRCCOPY);

        // クリッピング解除
        ::SelectClipRgn(m_pBackBuf->GetDC(), NULL);

        FPSTimer().DrawFPS(m_pBackBuf->GetDC());

        // バックバッファからディスプレイへ転送
        HDC   hDestDC = ::GetDC(GameApp().GetHWnd());
        ::BitBlt(hDestDC, 0, 0, m_pBackBuf->GetWidth(), m_pBackBuf->GetHeight(),
                 m_pBackBuf->GetDC(), 0, 0, SRCCOPY);
        ::ReleaseDC(GameApp().GetHWnd(), hDestDC);
    }

    return 0;
}
```