オブジェクト指向とゲームプログラミング

▶ DirectX Graphics編 － 第５回　WM_PAINTメッセージと再描画 ◀

再描画

あるウィンドウが、ほかのウィンドウによって隠された場合、その部分の画像は見えなくなります。覆っていたウィンドウが移動したり閉じられると、隠された部分が見えるようになり、その部分の再描画が必要になります。再描画が必要な領域を「無効な領域(Invalid Rectangle)」と呼びます。

無効な領域の再描画は、Windowsは行わないので、アプリケーションが行わなければなりません。無効な領域が発生すると、WindowsはWM_PAINTメッセージを発生させ、該当するウィンドウに再描画が必要なことを通知します。このメッセージは、正しく処理されるまでメッセージキューから取り除かれません。取り除かれない場合、WM_PAINTメッセージの処理しか行えなくなります。

WM_PAINTメッセージの処理

WM_PAINTメッセージを処理するには、最初にBeginPaint関数を呼び出します。BeginPaint関数を呼び出すと、再描画領域のデバイスコンテキストや座標などの情報が返されます。これらの情報をもとに再描画を行います。再描画が完了したら、EndPaint関数を呼び出します。EndPaint関数を呼び出すと、再描画を終えたことがWindowsに通知され、メッセージキューからWM_PAINTメッセージが取り除かれます。

```
// 再描画処理
case WM_PAINT:
    HDC         hDC;
    PAINTSTRUCT Paint;

    hDC = BeginPaint(hWnd, &Paint);     // 再描画開始

    // 再描画処理を行う
          ⋮

    EndPaint(hWnd, &Paint);             // 再描画終了

    return 0;
```

PAINTSTRUCT構造体には、再描画を行うのに必要な情報が格納されます。メンバには、再描画領域のデバイスコンテキスト(hdc)や再描画が必要な領域の座標(rcPaint)といったものがあります。

課　題

WM_PAINTメッセージを利用した再描画処理を追加しましょう。

(１)CGameAppクラスに、WM_PAINTメッセージを処理するOnPaint関数を追加します。次のプロトタイプを適切な場所に追加しましょう(アクセス指定は各自判断してください)。

```
    LRESULT OnPaint(const HWND hWnd, const WPARAM wParam, const LPARAM lParam);
```

(２)OnPaint関数の本体を実装します。以下のプログラムの足りない部分を補い、ソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                        WM_PAINTメッセージ処理                                       */
/*****************************************************************************************************/
LRESULT CGameApp::OnPaint(const HWND hWnd, const WPARAM wParam, const LPARAM lParam)
{
    PAINTSTRUCT    ps;
    ::??????????(hWnd, &ps);    // 再描画開始

    ::????????(hWnd, &ps);      // 再描画終了

    return 0;
}
```

(３)CGameApp::WindowProc関数を変更し、WM_PAINTメッセージを処理するようにします。以下のプログラム適切な場所に追加し、OnPaint関数が呼ばれるようにしましょう。

```
      case WM_PAINT:  return OnPaint(hWnd, wParam, lParam);
```

(４)CDXGraphics9クラスに、WM_PAINTメッセージが発生したときに呼び出してもらうOnPaint関数を作成します。以下のプロトタイプを適切な場所に追加しましょう。

```
    void OnPaint(const PAINTSTRUCT& inPaint);
```

DirectX Graphicsを用いて描画を行う場合、画面に描画されている画像はDirectX Graphicsが管理しています。そのため、WM_PAINTメッセージを受け取ったアプリケーションが再描画に必要な情報を取得し、それをCDXGraphics9::OnPaint関数に渡すことによって再描画を行うという設計にしています。

(５)OnPaint関数を実装します。以下のプログラムをソースファイルに追加しましょう。

```
/*****************************************************************************************************/
/*                                        WM_PAINTメッセージ処理                                       */
/*****************************************************************************************************/
void CDXGraphics9::OnPaint(const PAINTSTRUCT& inPaint)
{
    if(m_pD3DDevice == NULL)
        return;

    // 画面に描画されている画像を保持する領域(バックバッファ)を取得する
    IDirect3DSurface9*   pBackBuffer;
    m_pD3DDevice->GetBackBuffer(0, 0, D3DBACKBUFFER_TYPE_MONO, &pBackBuffer);

    HDC   hSrcDC;
    pBackBuffer->GetDC(&hSrcDC);        // バックバッファのデバイスコンテキストを取得する

    // 再描画しなければならない領域をクライアント領域に転送する
    ::BitBlt(inPaint.hdc, inPaint.rcPaint.left, inPaint.rcPaint.top,
             inPaint.rcPaint.right - inPaint.rcPaint.left, inPaint.rcPaint.bottom - inPaint.rcPaint.top,
             hSrcDC, inPaint.rcPaint.left, inPaint.rcPaint.top, SRCCOPY);

    pBackBuffer->ReleaseDC(hSrcDC);     // バックバッファのデバイスコンテキストを解放する
    pBackBuffer->Release();             // バックバッファの参照を解放する
}
```

(６)以下のプログラムをCGameApp.cppの適切な場所に追加し、CDXGraphics9::OnPaint関数が適切に呼び出されるようにしましょう。

```
    DXGraphics().OnPaint(ps);
```