第７章　書類印刷入門

ポイント：書類の出力方法を学ぶ。

１．プリント出力のメカニズム

２．画面表示とプリント出力の違い

３．文字と図形のプリント

7.1 はじめに

本章では、印刷に関する基本的な処理について解説する。最初に、ユーザーから印刷要求があった後、どのような仕組みで必要な書類が印刷されるかについて学ぶ。次に、図形や文字を出力する際、画面と紙への出力にどのような違いがあるか、また、どのようなことに注意すべきかについて解説する。

7.2 書類印刷の構造

最初に、ユーザーからの印刷要求があった後、どのような仕組みで書類が出力されるのかについて学ぶ。SPACE プレゼンターでの出力要求は、グラフや透視図などが描かれたウインドウで、マウス右ボタンをクリックしてプルダウンメニューを表示させることから始まる。次に、図 7-1 に示すメニューの中の図形出力を選択することで、出力要求メッセージが発せられる。

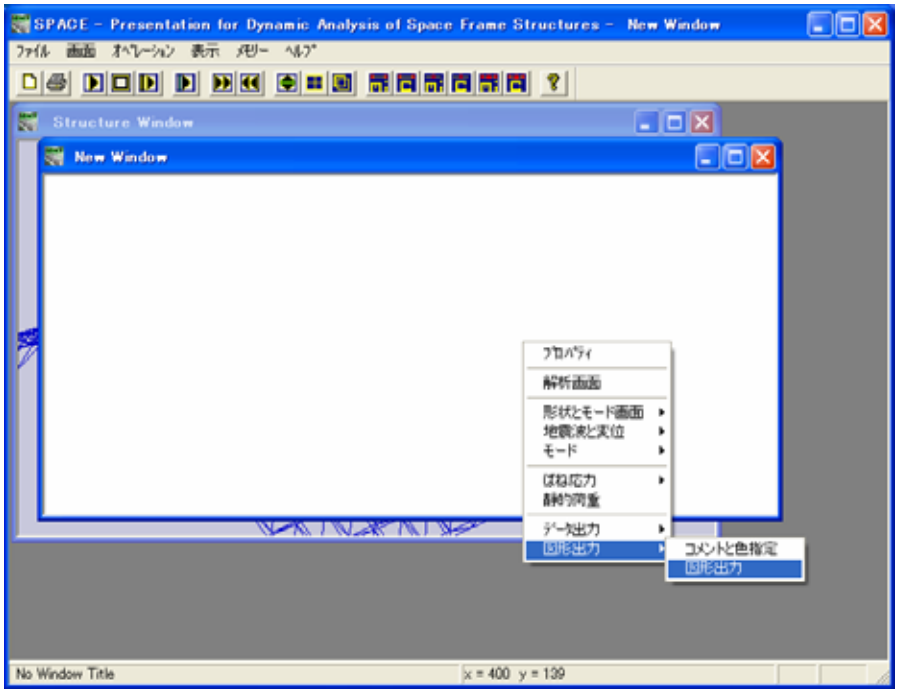

図 7-1 プルダウンメニューによる印刷書類の出力要求

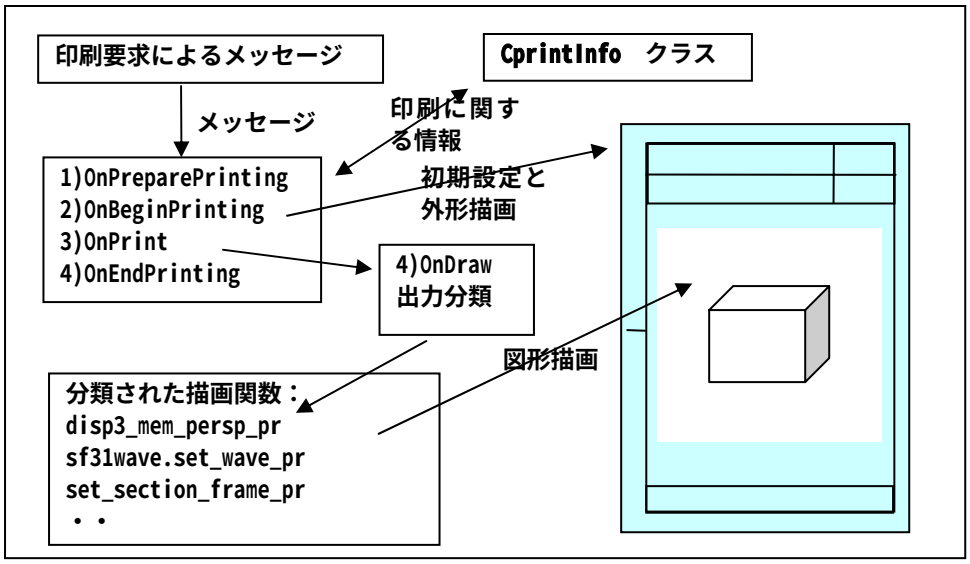

図 7-2 印刷システムの仕組み

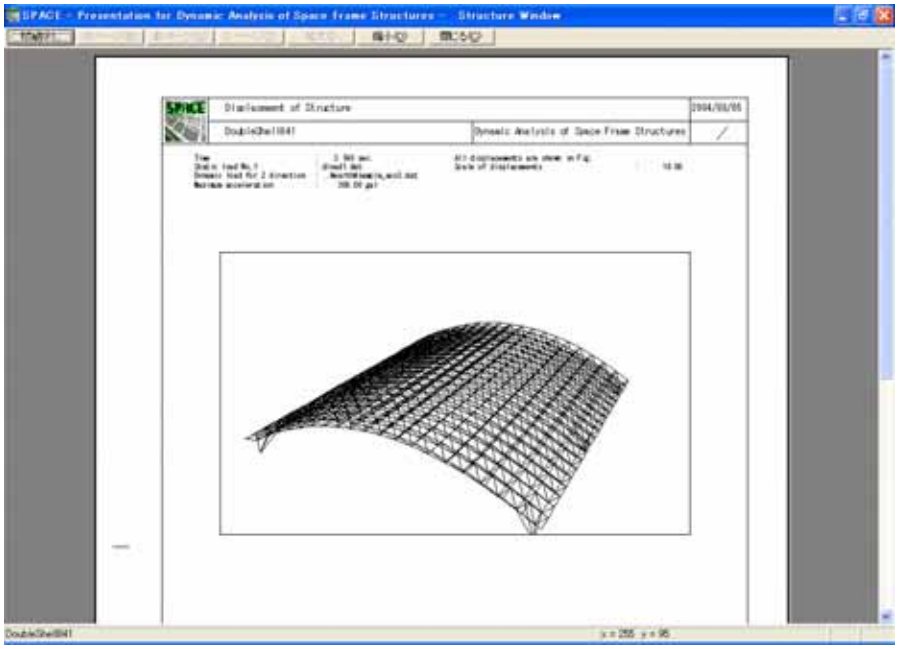

図 7-3 印刷書類のプレビュー画面

この出力要求は、次のメッセージマップ内の太文字で示すメッセージ ID_FILE_PRINT により、標準の印刷コマンドが実行される。実行される関数は次の順序である。

```
OnPreparePrinting()
OnBeginPrinting()
OnPrint()
OnDraw()
```

まず、クラス CSf31View のメッセージマップの印刷に関連する部分を

以下に示す。

```
//    ――――――――――――――――――――――――――――――――
//    ●    ウインドウメッセージマップ
//    ――――――――――――――――――――――――――――――――
IMPLEMENT_DYNCREATE(CSf31View, CView)
BEGIN_MESSAGE_MAP(CSf31View, CView)
        //{{AFX_MSG_MAP(CSf31View)
        ON_WM_LBUTTONDOWN()
        ON_WM_MOUSEMOVE()
        ・
        ・
        //}}AFX_MSG_MAP
        // 標準印刷コマンド
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
        ON_MESSAGE(IDS_MESSAGE_WIND, OnMessageWind)
END_MESSAGE_MAP()
```

先に示した関数に従って、実際のコードを以下に示す。最初の関数は、印刷の前処理を行う OnPreparePrinting()である。この関数の第１引数はクラス CprintInfo であり、印刷を制御するメンバー変数やメンバー関数を多く有している。精度の高い印刷処理を要求される場合は、このクラスを十分に理解することが必要となる。ここでは、メンバー関数 pInfo->SetMinPage(0)を用いて、ページ数の設定が行われている。

次は、実際の印刷前処理を行う関数 OnBeginPrinting()であるが、プレゼンターでは何も処理を行っていない。必要としない場合は、このように何もしなくても良い。同様に、次の関数 OnEndPrinting()も何も処理を行っていない。本来は、印刷処理が終了した後、この関数が呼ばれ、後始末を行うことになる。

```
//    ――――――――――――――――――――――――――――――――
//    ●    CSf31View クラスの印刷の前処理
//    ――――――――――――――――――――――――――――――――
//
BOOL CSf31View::OnPreparePrinting(CPrintInfo* pInfo)
{
//――――――――――――――――――――――――――――  ● デフォルトの印刷準備
      pInfo->SetMinPage(0);
      pInfo->SetMaxPage(0);
      return DoPreparePrinting(pInfo);
}
//    ――――――――――――――――――――――――――――――――
//    ●    CSf31View クラスの印刷の処理開始
//    ――――――――――――――――――――――――――――――――
//
void CSf31View::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
```

```
}
//    ――――――――――――――――――――――――――――――――
//    ●    印刷後の後処理
//    ――――――――――――――――――――――――――――――――
//
void CSf31View::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{

}
```

7.3 関数 OnPrint

本節では、実際の印刷処理を行う関数 OnPrint()について説明を行う。この関数では、印刷のための初期設定を行い、フォントやペンの設定を行う。特に、SPACE では印刷のための描画範囲を設定し、書類のヘッダーやフッター、あるいはタイトルなどを印字する。ここで特に大切なのは、ページプリンターの能力を知り、縦と横のドット数を取得することである。画面とページプリンターの１インチ当たりのドット数の比を知ることで、図形の描画倍率を得ることになる。

以下に示す実際のコードを観察することで、上記の処理を理解しよう。

```
//    ――――――――――――――――――――――――――――――――
//    ●    CSf31View クラスの印刷
//    ――――――――――――――――――――――――――――――――
//
void CSf31View::OnPrint(CDC* pDC, CPrintInfo* pInfo)
{
//――――――――――――――――――――――――――――――  ● ページプリンタの設定
    TEXTMETRIC tm;                                                    !1
    pDC->GetTextMetrics(&tm);                                         !2
    int cyChar = tm.tmHeight;
    rectPage = pInfo->m_rectDraw;                                     !3
    int iww = (rectPage.bottom - rectPage.top)/20;                    !4
    int ihh = (rectPage.right - rectPage.left)/20;
    int iwww =0;
    pDC->SelectObject ( &NewPenx_a[120]);
    FF_radio_color_pr = 0;                                  // color  zero reset
    if(F_radio_color_pr != 0 && m_pst_color != 0 ) FF_radio_color_pr = 1;
//――――――――――――――――――――――――――――――  ●  図形横の中央線を引く
    pDC->MoveTo ( rectPage.left, (rectPage.bottom + rectPage.top)/2);  !5
    pDC->LineTo ( rectPage.left+ihh/2, (rectPage.bottom + rectPage.top)/2);
//――――――――――――――――――――――――――――――  ●  ページ仕様セット
    rectPage.top +=   iww/2 ;                                         !6
    rectPage.bottom -=  iww/2;
    rectPage.right -=   ihh/2;
    rectPage.left +=   ihh + ihh/2;
//――――――――――――――――――――――――――――――  ●  フォントのセット
    int ipg=pInfo->m_nCurPage;
    fontbase= (rectPage.right-rectPage.left)/58;                      !7
```

```
    int point =fontbase;                                                    !8
    CClientDC dc(AfxGetMainWnd());
    prFont.CreateFont(point                                                 !9
                ,0,0,0,FW_NORMAL,FALSE,FALSE,0,
                SHIFTJIS_CHARSET,
                OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS,
                DEFAULT_QUALITY,
                DEFAULT_PITCH | FF_MODERN,
                "ＭＳ　明朝");
    point =fontbase*1.5;
    prFontx.CreateFont(point,0,0,0,FW_NORMAL,FALSE,FALSE,0,
                SHIFTJIS_CHARSET,
                OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS,
                DEFAULT_QUALITY,
                DEFAULT_PITCH | FF_MODERN,
                "ＭＳ　明朝");
                 point =fontbase;
    prFontb.CreateFont(point,0,900,0,FW_NORMAL,FALSE,FALSE,0,
                SHIFTJIS_CHARSET,
                OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS,
                DEFAULT_QUALITY,
                DEFAULT_PITCH | FF_MODERN,
                "ＭＳ　明朝");
                 point =fontbase*0.8;
    prFontxx.CreateFont(point,0,0,0,FW_NORMAL,FALSE,FALSE,0,
                SHIFTJIS_CHARSET,
                OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS,
                DEFAULT_QUALITY,
                DEFAULT_PITCH | FF_MODERN,
                "Times New Roman");
    CFont* pOldFont = (CFont*) pDC->SelectObject(&prFont);
//――――――――――――――――――――――――――――――― ●　ページ枠描画
    pDC->Rectangle(rectPage.left,rectPage.top ,rectPage.right,rectPage.bottom);  !10
    pDC->Rectangle(rectPage.left,rectPage.top ,rectPage.right,rectPage.top + iww);
    pDC->MoveTo ( rectPage.left+iww , rectPage.top+iww/2);
    pDC->LineTo ( rectPage.right, rectPage.top+iww/2);
    if(FF_radio_color_pr == 1) {                                            !11
        pDC->SelectObject ( &NewPenx_a[106]);
    }else{
        pDC->SelectObject ( &NewPenx_a[120]);
    }
    pDC->Rectangle(rectPage.left,rectPage.top ,rectPage.left+iww ,rectPage.top + iww); !12
    pDC->SelectObject ( &NewPenx_a[120]);
//――――――――――――――――――――――――――――――― ●　ページヘッダー出力
    CTime timex = CTime::GetCurrentTime();                                  !13
    CString datetime;
    datetime = timex.Format("%Y/%m/%d");
    pDC->TextOut(rectPage.left+(rectPage.right-rectPage.left)*0.905,
                        rectPage.top+fontbase*0.5,datetime);
    pDC->TextOut(rectPage.left+(rectPage.right-rectPage.left)*0.11,
```

```
                        rectPage.top+iww/2+fontbase*0.5,F_title);
    pDC->TextOut(rectPage.left+(rectPage.right-rectPage.left)*0.535,
              rectPage.top+iww/2+fontbase*0.5,"Dynamic Analysis of Space Frame Structures");
    pDC->MoveTo ( rectPage.left+(rectPage.right-rectPage.left)*0.53, rectPage.top+iww/2);
    pDC->LineTo (rectPage.left+ (rectPage.right-rectPage.left)*0.53, rectPage.top+iww);
    pDC->MoveTo ( rectPage.left+(rectPage.right-rectPage.left)*0.9, rectPage.top);
    pDC->LineTo (rectPage.left+ (rectPage.right-rectPage.left)*0.9, rectPage.top+iww);
    pDC->MoveTo ( rectPage.left+(rectPage.right-rectPage.left)*0.965, rectPage.top+iww*0.6);
    pDC->LineTo ( rectPage.left+(rectPage.right-rectPage.left)*0.945, rectPage.top+iww*0.9);
//――――――――――――――――――――――――――――――― ●　ページフッター出力
    F_titl_xichi = rectPage.left+(rectPage.right-rectPage.left)*0.11;          !14
    F_titl_yichi = rectPage.top+fontbase*0.5;
    pDC->Rectangle(rectPage.left,rectPage.bottom-2*iww ,rectPage.right,rectPage.bottom);
    int base= (rectPage.right-rectPage.left)/60;
    Crect  rectx(rectPage.left+base,rectPage.bottom-2*iww+base*0.2,
                rectPage.right-base,rectPage.bottom-base*0.2);
    pDC->DrawText(m_coment,rectx,DT_LEFT | DT_WORDBREAK );
    pDC->SelectObject(pOldFont);
//――――――――――――――――――――――――――――――― ●　印刷書類右上のビットマップ描画
    CBitmap* bitmap = new CBitmap;                                          !15
    if(FF_radio_color_pr == 1) {
      bitmap->LoadBitmap(IDB_BITMAP1);
    }else{
      bitmap->LoadBitmap(IDB_BITMAP1);
    }
//――――――――――――――――――――――――――――――― ● 実際の図形などを描画する前の初期設定
    CDC* pMemDC = new CDC;                                                  !16
    pMemDC->CreateCompatibleDC(pDC);
    CBitmap *poldBitmap =(CBitmap*)pMemDC -> SelectObject(bitmap);
    pMemDC->SetMapMode(pDC->GetMapMode());
    pDC->StretchBlt(rectPage.left+fontbase*0.3,rectPage.top+fontbase*0.3,
                 iww-fontbase*0.6,iww-fontbase*0.6,pMemDC,0,0,32,32,SRCCOPY);
    delete pMemDC->SelectObject(poldBitmap);
    delete pMemDC;
//――――――――――――――――――――――――――――――― ● 描画範囲を指定する
    rectPage.top    +=   iww ;                                             !17
    rectPage.bottom -=  2*iww;
    rectPage.right  -=   ihh;
    rectPage.left   +=   ihh;
    pDC->SelectObject ( &NewPenx_a[120]);
    int screen_width = GetSystemMetrics(SM_CXSCREEN);
    F_xbai = (rectPage.right - rectPage.left)/screen_width;
    F_ixs=rectPage.left;
    F_iys=rectPage.top+iww/2;
    OnPrepareDC(pDC,pInfo) ;
//――――――――――――――――――――――――――――――― ●　下記の関数を用いて、図形や文字を印刷
    OnDraw(pDC);                                                            !18
}
```

先の手続きに関連して、上の関数を説明する。

1. フォント仕様 TEXTMETRIC の構造体 tm を生成する。

2. 現在のフォントの寸法情報を取りだし、フォントの高さを変数 cyChar にセットする。
3. 印刷書類の大きさを rectPage に取得する。
4. 印刷書類の縦と横の 1/20 を変数 iww と ihh に設定する。
5. 図形の中央左に、書類に綴じ穴を開けるときに利用する線分を印刷する。
6. 図形の描画可能領域を設定する。ここでは、上のヘッダー部分と左右、及び下部分の余白を取り除く。
7. タイトル用フォントの大きさであるフォントベースを書類の大きさから決定する。
8. フォントベースを変数 point にセットする。
9. 大きさの異なるプリント用フォント 4 種類を関数 CreateFont()を用いて生成する。
10. ヘッダー部分の枠を描画する。
11. カラー出力かどうかチェックし、ペンを選択する。
12. ページ枠を描く。
13. 印刷時刻を取得し、仕様と整えた後、印刷する。
14. 書類のフッターを印字する。
15. 書類右上のビットマップを入力する。
16. 入力した図形を書類の所定位置に転送し、印刷する。
17. 印刷書類の大きさ rectPage を再設定する。
18. 実際の図形描画などを関数 OnDraw()で行う。

7.4 関数 OnDraw による印刷

関数 OnDraw()は、第２章で説明したように、図形の再描画という重要な役割を担っている。ただし、この役割以外に実際の印刷処理を、特に図形に関する印刷処理の制御を行っている。まず、CSf31View クラスの関数 OnDraw()を再度見てみよう。ここでは、ウインドウ管理コードを取得した後、関数 pDC->IsPrinting()が記述されており、この関数によって印刷処理か、もしくは画面への描画かが分類される。この関数を含んで OnDraw()の構造を以下に示す。

```
//――――――――――――――――――――――――――――  ● プリンターによる描画
if(pDC->IsPrinting())
        m_x1_pr =0;
        m_x2_pr =0;
     •
```

```
      ・
{
//――――――――――――――――――――――――――――――― ● 画面における描画
}else{
        CSf31Doc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
      ・
      ・
}
```

上の構造を考慮しつつ、再度 CSf31View クラスの OnDraw 関数を観察する。

```
//    ――――――――――――――――――――――――――――――――――
//    ●    CSf31View クラスの子ウインドウの描画
//    ――――――――――――――――――――――――――――――――――
//
void CSf31View::OnDraw(CDC* pDC)
{
        HWND hwnd = this->GetSafeHwnd();                              !1
        if(IsWindow(hwnd) == FALSE) return;
        long fno_yx = GetWindowLong(hwnd,GWL_USERDATA);               !2
        int fno_xy  = getwindx(fno_yx);
        int fno_xxx = getwindy((long)fno_yx);
        if(fno_xy == -1) return;
        int ii=fno_yx-1;
//――――――――――――――――――――――――――――――― ● プリンターによる描画
if(pDC->IsPrinting())                                                 !3
{
        m_x1_pr =0;                                                   !4
        m_x2_pr =0;
        m_y_pr  =0;
        m_han_pr =0;
switch (fno_xy){                                                      !5
case STRUCT:
    if( F_time_ii > m_nstep && fno_xxx != 2) F_time_ii = m_nstep;
    if(F_time_ii <= 0   ){
      if( fno_xxx == 0) pre_disp_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 2) disp3_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 3) disp7_mem_persp_pr(hwnd,pDC);
      if( fno_xxx != 3 && fno_xxx != 1 && fno_xxx != 0 && fno_xxx != 2)  disp_mem_persp_pr(hwnd,pDC);
    }else{
      if( fno_xxx == 0) pre_disp_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 1 && F_time_ii <= m_nstep ) disp2_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 2) disp3_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 3) disp7_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 4 && F_time_ii <= m_nstep ) disp5_mem_persp_pr(hwnd,pDC);
      if( fno_xxx == 5 && F_time_ii <= m_nstep ) disp6_mem_persp_pr(hwnd,pDC);
    }
    break;
//    ――――――――――――――――――――――――――――――――――
//    ●    CSf31View クラスの描画
//     fno_xxx : 0  option
//     fno_xxx : 1  structure
```

```
//      fno_xxx : 2  mode displacements
//      fno_xxx : 3  maximum stresses
//      fno_xxx : 4  maximum response of displacement, velocity and acceleration
//      fno_xxx : 5  mode decomposition
//     ――――――――――――――――――――――――――――――――
//
case WAVE:                                                          !6
    if(m_dat_struct == 0)  sf31wave.set_wave_pr(pDC,hwnd);
    if(m_dat_struct == 1)  sf31ftt.set_frame_pr(pDC,hwnd);
    if(m_dat_struct == 2)  sf31shear.set_frame_pr(pDC,hwnd);
    break;
case SLOAD:                                                         !7
    sf31load.set_load_frame_pr(pDC,hwnd);
    break;
case STRESS:                                                        !8
    set_member_frame_pr(pDC,hwnd);
    break;
case MODEWD:                                                        !9
    sf31mode.set_mode_frame_pr(pDC,hwnd);
    break;
case DISMAX:                                                        !10
    sf31mode.set_mode_frame_pr(pDC,hwnd);
    break;
case SECTION:                                                       !11
    set_section_frame_pr(pDC,hwnd,F_time_ii);
    break;
default:
    break;
}
//――――――――――――――――――――――――――――  ● 画面における描画
}else{
        CSf31Doc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
      •
      •
}
}
```

先の手続きに関連して、上の関数を説明する。

１．アクティブなウインドウハンドル hWnd を取得する。
２．このウインドウハンドルを用いて、ウインドウ管理システムより、画面番号、ウインドウコード、構造図出力用コードを取得する。
３．関数 pDC->IsPrinting()で印刷処理かどうかチェックする。この関数の戻り値が TRUE の場合、以下の処理を行う。
４．印刷処理用の初期設定を行う。
５．印刷処理を switch 文を用いて分類する。ケース STRUCT では、構造図関連の印刷処理を行う。ここでは、アニメーションの時刻と構造図出力用コードによって、さらに印刷処理を分類する。

６．ケース WAVE では、地震波形などの履歴図関連の印刷処理を行う。ここではさらに、メンバー変数 m_dat_struct によって図形処理が分類される。
７．ケース SLOAD では、擬似的静的荷重の履歴が描画される。
８．ケース STRESS では、部材の応力状態が描画される。
９．ケース MODEWD では、構造物の振動数、減衰率などのグラフが印刷処理される。
10．ケース DISMAX では、構造物の最大変位や最大応力が透視図として描画され、印刷される。
11．ケース SECTION では、部材断面の応力状態や応力・ひずみの関係が描画され、印刷される。

7.5 実際の印刷処理の記述

本節では、実際に印刷処理を行う関数を用いて、画面表示と如何なる点が異なった処理なのか、どこに気を付けなければならないかについて学ぶ。図形処理の基本は、第２章、４章で既に述べられているので、十分に理解できていると思う。ここでは、構造物の自由振動モード図を印刷する処理を例として取り上げ、画面表示との違いを中心に、印刷処理を学ぶことにする。まず、関数 OnDraw()でコールされ、自由振動モード図を印刷する関数 disp3_mem_persp_pr()を以下に示す。

```
//     ―――――――――――――――――――――――――――――――――
//     ●    disp3_mem_persp_pr：振動モード図の印刷処理
//     ―――――――――――――――――――――――――――――――――
//
void CSf31View::disp3_mem_persp_pr(HWND hWnd,CDC* pDC)
{
//――――――――――――――――――――――――――――――  ●  印刷用フォントを選択
            char buff[200];
            CFont* pOldFont = (CFont*) pDC->SelectObject(&prFontxx);        !1
//――――――――――――――――――――――――――――――  ●  印刷用範囲を取得
            CRect rcFrame;
            ::GetClientRect(hWnd ,&rcFrame);                               !2
//――――――――――――――――――――――――――――――  ●  フォントの大きさから文字出力位置計算
            int iy=rectPage.top ;                                          !3
            int ix=rectPage.left ;
            int F_ipy = fontbase*0.9;                                      !4
            int x1= 12*fontbase;
            int icc=0;
            int iccc=0;
//――――――――――――――――――――――――――――――  ●  タイトル表示
            iy=iy-2*fontbase;                                              !5
            sprintf(buff,"Mode Displacement of Structure");
```

```
            pDC->SelectObject(&prFont );
            pDC->TextOut(F_titl_xichi,F_titl_yichi,buff);                    !6
//――――――――――――――――――――――――――――――― ●　表示時刻出力
            pDC->SelectObject(&prFontxx);
            iy=rectPage.top;                                                 !7
            iy=iy+F_ipy;                                                     !8
            sprintf(buff,"Time");
            pDC->TextOut(ix,iy,buff);
            sprintf(buff,": %8.3f sec.",F_Time);
            pDC->TextOut(ix+x1,iy,buff);
            icc=icc+1;
//――――――――――――――――――――――――――――――― ●　表示モード番号
            iy=iy+F_ipy;                                                     !9
            sprintf(buff,"Mode number");
            pDC->TextOut(ix,iy,buff);
            sprintf(buff,": %4d ",m_dat_mode_number);
            pDC->TextOut(ix+x1,iy,buff);
            icc=icc+1;
//――――――――――――――――――――――――――――――― ●　振動数
            iy=iy+F_ipy;
            sprintf(buff,"Frequency of mode");
            pDC->TextOut(ix,iy,buff);
            sprintf(buff,": %8.3f 1/sec. ",sf31mode.datget_omeg());
            pDC->TextOut(ix+x1,iy,buff);
            icc=icc+1;
//――――――――――――――――――――――――――――――― ●　固有周期
            iy=iy+F_ipy;
            sprintf(buff,"Natural period");
            pDC->TextOut(ix,iy,buff);
            sprintf(buff,": %8.3f sec. ",sf31mode.datget_t());
            pDC->TextOut(ix+x1,iy,buff);
            icc=icc+1;
//――――――――――――――――――――――――――――――― ●　節点変位の表示オプション
            ix=rectPage.left+(rectPage.right - rectPage.left)*0.5;
            iy=rectPage.top;
            if( m_pst_disp == 0){
              iy=iy+F_ipy;
              sprintf(buff,"All displacements are shown in Fig.");
              pDC->TextOut(ix ,iy,buff);
              iccc=iccc+1;
            }
            if( m_pst_disp == 1){
              iy=iy+F_ipy;
              sprintf(buff,"Displacements for X direction are shown in FIg.");
              pDC->TextOut(ix ,iy,buff);
              iccc=iccc+1;
            }
            if( m_pst_disp == 2){
              iy=iy+F_ipy;
              sprintf(buff,"Displacements for Y direction are shown in Fig.");
              pDC->TextOut(ix ,iy,buff);
              iccc=iccc+1;
            }
            if( m_pst_disp == 3){
```

```
                iy=iy+F_ipy;
                sprintf(buff,"Displacements for Z direction are shown in Fig.");
                pDC->TextOut(ix ,iy,buff);
                iccc=iccc+1;
              }
              if( m_pst_disp == 4){
                iy=iy+F_ipy;
                sprintf(buff,"NO displacement are shown in Fig.");
                pDC->TextOut(ix ,iy,buff);
                iccc=iccc+1;
              }
//――――――――――――――――――――――――――――――  ●　部材の描画オプション
              if(Dlg_pstruct.m_check_mem_used){
              iy=iy+F_ipy;
              sprintf(buff,"Member displacements as follows are shown");
              pDC->TextOut(ix ,iy,buff);
              iy=iy+F_ipy;
              sprintf(buff,"[ %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d  %4d ] "
                     ,Dlg_pstruct.m_edit_mem11,Dlg_pstruct.m_edit_mem22,
                     Dlg_pstruct.m_edit_mem33,Dlg_pstruct.m_edit_mem44,Dlg_pstruct.m_edit_mem55
                     ,Dlg_pstruct.m_edit_mem66,Dlg_pstruct.m_edit_mem77,Dlg_pstruct.m_edit_mem88
                     ,Dlg_pstruct.m_edit_mem99,Dlg_pstruct.m_edit_mem10);
              pDC->TextOut(ix,iy,buff);
              iccc=iccc+2;
          }
//――――――――――――――――――――――――――――――  ●　図形の描画範囲設定
              if(iccc > icc) icc=iccc;
              rectPage.top += F_ipy*icc;
              int iww,iwh;
              iww = (rectPage.right - rectPage.left) ;                          !10
              iwh = (rectPage.bottom - rectPage.top) ;
//――――――――――――――――――――――――――――――  ●　図形倍率計算とその範囲セット
              float bairitu=2;
              bairitu =  setpos(iww,iwh,rcFrame.right,rcFrame.bottom);          !11
              disp_frame_w_pr(pDC);
              CRect rectPage_g;
              rectPage_g.left =F_pos_pr[0][0] ;
              rectPage_g.top =F_pos_pr[1][0];
              rectPage_g.right =F_pos_pr[0][1];
              rectPage_g.bottom =F_pos_pr[1][1];
//――――――――――――――――――――――――――――――  ●　図形描画範囲をクリップ
              pDC->IntersectClipRect(rectPage_g);                               !12
//――――――――――――――――――――――――――――――  ●　選択振動モードを入力
              float amp = sf31mode.datset_mode(F_Time,F_scalep[3]);
//――――――――――――――――――――――――――――――  ●　節点位置とモードを重ねて、透視変換
              TOSHIMPR(sf31mode.FF_mode,posit,&node,&m_rr[0][0],
                       F_bz,&F_pos_pr[0][0],&F_amx,&F_amy,
                       iconsb,&mnbsb,imeme,&amp,&m_base[0],&bairitu,&m_pst_disp); !13
//――――――――――――――――――――――――――――――  ●　図形描画処理
              disp_graph_1_pr(pDC,bairitu);                                     !14
//――――――――――――――――――――――――――――――  ●　クリップ解除
              pDC->SelectClipRgn(NULL);                                         !15
}
```

上記関数で、特に印刷処理に関連する箇所について説明する。

1. 印刷用のフォントを選択する。
2. このウインドウハンドルを用いて、ウインドウの描画範囲を取得する。
3. 印刷印字位置の初期設定を行う。
4. 印刷用フォントをベースにした印字間隔 F_ipy を設定する。
5. 印字位置を移動し、フォントを変更する。
6. タイトルを出力する。
7. 時刻を出力するために、初期位置を設定する。
8. ｙ方向位置を移動し、時刻を出力する。
9. ｙ方向位置を移動し、モード番号を出力する。その後、位置を設定して、各種の情報を印字する。
10. 印刷用紙の描画範囲である幅と高さをセットする。
11. 関数 setpos()を用いて、印刷用紙の描画範囲と画面ウインドウの描画範囲を比較して、印刷の描画倍率を求める。
12. 関数 pDC->IntersectClipRect(rectPage_g)を用いて、印刷範囲をクリップする。これで、この範囲以外の図形描画は無視される。
13. 節点位置と選択したモードを重ねて、透視変換を行う。
14. 関数 disp_graph_1_pr()を用いて、図形を印刷処理する。
15. クリップを解除する。

次に、背景処理を行う関数 disp_frame_w_pr()と、描画倍率を求める関数setpos()、図形処理を行う関数disp_graph_1_pr()を具体的に示す。ただし、これらの関数は、既に説明した関数の応用であり、コメント行を参考にすれば理解するのは容易である。プログラムを理解することは、読者の課題とする。

```
//    ――――――――――――――――――――――――――――――――
//    ●    disp_frame_w_pr:背景印刷処理
//    ――――――――――――――――――――――――――――――――
//
void CSf31View::disp_frame_w_pr(CDC* pDC)
{
//――――――――――――――――――――――――――――  ●  カラー出力か
        if(FF_radio_color_pr != 0  ){
            int i1,i2,i3;
//――――――――――――――――――――――――――――  ●  背景描画位置セット
            CRgn theRgn;
            theRgn.CreateRectRgn (F_pos_pr[0][0],F_pos_pr[1][0],
            F_pos_pr[0][1],F_pos_pr[1][1]);
```

```
//――――――――――――――――――――――――――――――――  ●　背景をブラシで塗り、境界を描く
            CBrush MyBrush(Brash_color);
            CBrush* pOldBrush = pDC->SelectObject(&MyBrush);
            pDC->FillRgn (&theRgn,&MyBrush);
            pDC->SelectObject(pOldBrush);
            MyBrush.DeleteObject();
            CPen NewPen (PS_SOLID,3,RGB(255,0,0));
            CPen* pOriginalPen = pDC->SelectObject ( &NewPen);
            pDC->MoveTo (  F_pos_pr[0][0]-fontbase*0.4, F_pos_pr[1][0]-fontbase*0.4);
            pDC->LineTo (  F_pos_pr[0][1]+fontbase*0.4, F_pos_pr[1][0]-fontbase*0.4);
            pDC->LineTo (  F_pos_pr[0][1]+fontbase*0.4, F_pos_pr[1][1]+fontbase*0.4);
            pDC->LineTo (  F_pos_pr[0][0]-fontbase*0.4,  F_pos_pr[1][1]+fontbase*0.4);
            pDC->LineTo (  F_pos_pr[0][0]-fontbase*0.4,  F_pos_pr[1][0]-fontbase*0.4);
            pDC->SelectObject ( pOriginalPen);
        }else{
//――――――――――――――――――――――――――――――――  ●　白黒印刷であり、境界を描く
            CPen NewPen (PS_SOLID,1,RGB(0,0,0));
            CPen* pOriginalPen = pDC->SelectObject ( &NewPen);
            pDC->MoveTo (  F_pos_pr[0][0], F_pos_pr[1][0]);
            pDC->LineTo (  F_pos_pr[0][1], F_pos_pr[1][0]);
            pDC->LineTo (  F_pos_pr[0][1], F_pos_pr[1][1]);
            pDC->LineTo ( F_pos_pr[0][0],  F_pos_pr[1][1]);
            pDC->LineTo (  F_pos_pr[0][0],  F_pos_pr[1][0]);
            pDC->SelectObject ( pOriginalPen);
        }
}
//    ――――――――――――――――――――――――――――――――――――
//    ●    setpos:印刷用倍率計算
//    ――――――――――――――――――――――――――――――――――――
//
float CSf31View::setpos(int iww,int iwh,int right,int bottom)
{
        float a1=1.;
        float bairitu =1.;
        float aww = iww*0.9;
        float awh = iwh*0.7;
//――――――――――――――――――――――――――――――――  ●　画面と印刷範囲の比から倍率計算
        if(right != 0) bairitu = float(aww)/float(right);
   if(bottom != 0 ) a1= float(awh)/float(bottom);
        if(a1 < bairitu )  bairitu = a1;
//――――――――――――――――――――――――――――――――  ●　印刷書類の中心位置計算
        float center =(rectPage.left+rectPage.right)*0.5;
//――――――――――――――――――――――――――――――――  ●　印刷範囲セット
         F_pos_pr[0][0]=center - right*bairitu*0.5;
         F_pos_pr[1][0]=rectPage.top + iwh*0.1;
         F_pos_pr[0][1]=center + right*bairitu*0.5;
         F_pos_pr[1][1]=rectPage.top + iwh*0.1 + bottom*bairitu;
        return bairitu;
}
//    ――――――――――――――――――――――――――――――――――――
//    ●    disp_graph_1_pr:モード図及びオプション画面の印刷処理
//    ――――――――――――――――――――――――――――――――――――
//
void CSf31View::disp_graph_1_pr(CDC* pDC,float bairit)
```

```
{
//――――――――――――――――――――――――――――――――  ●　ペンの選択
            CPen NewPen;
            if(FF_radio_color_pr == 0){
              NewPen.CreatePen (PS_SOLID,1,RGB(0,00,0));
            }else{
              NewPen.CreatePen (PS_SOLID,F_edit_line_width_g,RGB(255,255,255));
            }
//――――――――――――――――――――――――――――――――  ●　集中質量表示
            CPen* pOriginalPen = pDC->SelectObject ( &NewPen);
            int i1,i2,i3;
            int ix_f,iy_f;
            if(m_pst_mass != 0){
             int bdn5=bairit*5;
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[123]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[107]);
             for(int i = 0; i < node; i++){
                   int iik=node;
              if(F_mass[i+iik] != 0){
                i1 = F_bz[i*2];
                i2 = F_bz[i*2+1];
                pDC->Arc(i1-bdn5,i2-bdn5,i1+bdn5,i2+bdn5,i1-bdn5,i2,i1-bdn5,i2);
                  }}
            }
//――――――――――――――――――――――――――――――――  ●　構造透視図：部材表示オプションなし
            if(m_pst_options == 0){
//――――――――――――――――――――――――――――――――  ●　部材グループ部材表示あり
             if(m_radio_mem == 1 ){
              for( int i = 0; i < mnbsb; i++){
               i3 = i*2;
               i1 = (iconsb[i3]-1)*2;
               i2 = (iconsb[i3+1]-1)*2;
               if(imeme_line[i] == m_prop_mem){
                if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[123]);
                if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[106]);
                    }else{
                pDC->SelectObject ( &NewPen);
                    }
               pDC->MoveTo ( (int)F_bz[i1], (int)F_bz[i1+1] );
               pDC->LineTo ( (int)F_bz[i2], (int)F_bz[i2+1] );
                    }
//――――――――――――――――――――――――――――――――  ●　部材グループ部材表示なし
            }else{
             for( int i = 0; i < mnbsb; i++){
              i3 = i*2;
              i1 = (iconsb[i3]-1)*2;
              i2 = (iconsb[i3+1]-1)*2;
              pDC->MoveTo ( (int)F_bz[i1], (int)F_bz[i1+1] );
              pDC->LineTo ( (int)F_bz[i2], (int)F_bz[i2+1] );
             }
            }
           }
//――――――――――――――――――――――――――――――――  ●　構造透視図：部材表示オプションあり
            if(m_pst_options == 1){
```

```
            int mem,mem_line;
//――――――――――――――――――――――――――――――  ●  部材グループ部材表示あり
            if(m_radio_mem == 1){
             for( int i = 0; i < mnbsb; i++){
              i3 = i*2;
              i1 = (iconsb[i3]-1)*2;
              i2 = (iconsb[i3+1]-1)*2;
              mem_line = imeme_line[i];
              mem = imeme[i];
              if(m_pst_line[mem_line-1] == 1){
               if(mem == m_prop_mem){
                if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[123]);
                if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[106]);
              }else{
                 pDC->SelectObject ( &NewPen);
              }
              pDC->MoveTo ( (int)F_bz[i1], (int)F_bz[i1+1] );
                pDC->LineTo ( (int)F_bz[i2], (int)F_bz[i2+1] );
               }}
//――――――――――――――――――――――――――――――  ●  部材グループ部材表示なし
              }else{
               for( int i = 0; i < mnbsb; i++){
               mem_line = imeme_line[i];
               mem = imeme[i];
               if(m_pst_line[mem_line-1] == 1){
                i3 = i*2;
                i1 = (iconsb[i3]-1)*2;
                i2 = (iconsb[i3+1]-1)*2;
                pDC->MoveTo ( (int)F_bz[i1], (int)F_bz[i1+1] );
                pDC->LineTo ( (int)F_bz[i2], (int)F_bz[i2+1] );
               }}
              }
             }
//――――――――――――――――――――――――――――――  ●  塑性ヒンジ表示
            if(m_dat_struct == 3  ){
              disp_graph_8_pr( pDC,bairit) ;
            }
//――――――――――――――――――――――――――――――  ●  境界条件表示
            if(m_radio_bound == 1){
            for(int i = 0; i < node; i++){
             if(F_bound[i] != 0){
              ix_f = F_bz[i*2];
              iy_f = F_bz[i*2+1];
              int bdn5=bairit*5;
              int bdn8=bairit*8;
              int bdn11=bairit*11;
              if(F_bound[i] == 1){
               if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[123]);
               if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[105]);
                pDC->MoveTo ( ix_f-bdn5, iy_f-bdn5 );
                pDC->LineTo ( ix_f+bdn5, iy_f-bdn5  );
                pDC->LineTo ( ix_f+bdn5, iy_f+bdn5  );
                pDC->LineTo ( ix_f-bdn5, iy_f+bdn5  );
                pDC->LineTo ( ix_f-bdn5, iy_f -bdn5 );
```

```
             }
             if(F_bound[i] == 2){
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[105]);
             pDC->MoveTo ( ix_f, iy_f -bdn5);
             pDC->LineTo ( ix_f-bdn5, iy_f+bdn5  );
             pDC->LineTo ( ix_f+bdn5, iy_f+bdn5  );
             pDC->LineTo ( ix_f, iy_f-bdn5);
            }
            if(F_bound[i] == 3){
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[105]);
             pDC->MoveTo ( ix_f, iy_f-bdn5 );
             pDC->LineTo ( ix_f-bdn5, iy_f+bdn5  );
             pDC->LineTo ( ix_f+bdn5, iy_f+bdn5  );
             pDC->LineTo ( ix_f, iy_f-bdn5);
             pDC->MoveTo ( ix_f-bdn5, iy_f+bdn8 );
             pDC->LineTo ( ix_f+bdn5, iy_f +bdn8 );
            }
            if(F_bound[i] == 4){
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[105]);
             pDC->MoveTo ( ix_f, iy_f-bdn5 );
             pDC->LineTo ( ix_f-bdn5, iy_f+bdn5  );
             pDC->LineTo ( ix_f+bdn5, iy_f+bdn5  );
             pDC->LineTo ( ix_f, iy_f-bdn5);
             pDC->MoveTo ( ix_f-bdn5, iy_f+bdn8 );
             pDC->LineTo ( ix_f+bdn5, iy_f +bdn8 );
             pDC->MoveTo ( ix_f-bdn5, iy_f+bdn11 );
             pDC->LineTo ( ix_f+bdn5, iy_f +bdn11 );
            }
            if(F_bound[i] == 5){
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[103]);
             pDC->Arc(ix_f-bdn5,iy_f-bdn5,ix_f+bdn5,iy_f+bdn5,ix_f-bdn5,iy_f,ix_f-bdn5,iy_f);
            }
            }}}
//――――――――――――――――――――――――――――――  ●  節点荷重表示
            if(m_radio_load != 0){
              if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
              if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[106]);
              for(int i = 0; i < node; i++){
               for(int j=0;j < 3; j++){
                disp_load_pr(pDC,i,j,m_radio_load,bairit);
               }
              }
             }
//――――――――――――――――――――――――――――――  ●  部材の主軸表示
            if(m_radio_cood != 0){
             for(int i = 0; i < mnbsb; i++){
              disp_cood_pr(pDC,i,bairit);
             }
            }
//――――――――――――――――――――――――――――――  ●  質量表示
```

```
            if(m_radio_mass != 0){
             int bdn5=bairit*5;
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[106]);
             if(m_radio_mass == 2) {
             if(FF_radio_color_pr == 0) pDC->SelectObject ( &NewPenx_a[121]);
             if(FF_radio_color_pr == 1) pDC->SelectObject ( &PrPenx_a[109]);
            }
            for(int i = 0; i < node; i++){
              int iik=node*(m_radio_mass-1);
              if(F_mass[i+iik] != 0){
                ix_f = F_bz[i*2];
                iy_f = F_bz[i*2+1];
               pDC->Arc(ix_f-bdn5,iy_f-bdn5,ix_f+bdn5,iy_f+bdn5,ix_f-bdn5,iy_f,ix_f-bdn5,iy_f);
             }}
           }
           pDC->SelectObject ( pOriginalPen);
         }
 }
```