Chapter 48 – Basic Microsoft Windows: Graphics

The first steps to basic graphics in a window is very similar to those in chapter 47 where a dialog box based program was created. Like the previous chapter, the instructions in this section are based on using Microsoft Visual C++ 5.0 and Microsoft Developer Studio 97. This example program will draw a graph of the Mandelbrot Set in a window document.

Basic Graphics

In order to draw, pens of various colors need to be created. Pens are objects of class CPen that are created by a call to a constructor with the following form:

\[ \text{CPen(style, thickness, color);} \]

where style is an expression or value indicating whether the pen will draw a solid line or some other style
thickness is an integer expression or value 1 or greater that sets the thickness in pixels of the line the pen draws
color is the return of a call to the function RGB

The function RGB accepts three integer expressions for color values for red, green and blue. Each of these values can be from 0 (lowest intensity) to 255 (highest intensity). Function RGB returns the color given by mixing the three colors. Function RGB has the form:

\[ \text{RGB(expression, expression, expression);} \]

Here is an example of declaring a pen:

\[ \text{CPen penRed(PS_SOLID, 1, RGB(255,0,0));} \]

This example creates a pen that can draw solid lines one pixel wide of the color bright red.

The function that uses objects of class CPen will have a pointer variable passed to it created named pDC of class CDC, which will represent the document that a window is displaying. It is through pDC that the pen can be manipulated.

A call to the CDC member function SelectObject selects the pen that is passed as the parameter to be used by any drawing commands. For example:

\[ \text{PDC->SelectObject(&penRed);} \]
Drawing a line with the selected pen is accomplished with the \textit{CDC} member function \textit{LineTo}, which accepts a target (X,Y) pair of pixel coordinates, X being across and Y being down from the upper left corner. \textit{LineTo} will draw a line from wherever the pen is currently located to the target pair. Here is its form:

\begin{verbatim}
CDC-object->LineTo(int-expression, int-expression);
\end{verbatim}

For example:

\begin{verbatim}
pDC->LineTo(100,300);
\end{verbatim}

The \textit{CDC} member function \textit{MoveTo} lets the pen move to a new target without drawing a line. Like \textit{LineTo}, \textit{MoveTo} accepts a target (X,Y) pair of pixel coordinates. For example:

\begin{verbatim}
pDC->MoveTo(50,73);
\end{verbatim}

There are many other graphics functions that will not be discussed here. They are easily available via the Help function of the Visual C++ environment.

\section*{Steps in Making a Single Document Windowing Program}

1. select the \textit{File} pull down menu, then select the \textit{New} option
2. in the dialog box, select the \textit{Projects} tab
3. select the \textit{mfcAppWizard(exe)}, type in a name (\textit{Mandelbrot} for this example), select the location for the project and click on the \textit{OK} button
4. select \textit{Single}, then press \textit{Next}
5. select \textit{Next}
6. select \textit{Next}
7. turn off all controls except the \textit{3d Controls}, then select \textit{Finish}

\section*{Coding the Example}

Using the \textit{File View} of the workspace partition, open \textit{Source}. Next, open the file \textit{MandelView.cpp} and find the \textit{OnDraw} function. Insert \texttt{#include <math.h>} just before function \texttt{CMandelbrotView::OnDraw}.

\begin{verbatim}
#include <math.h>

void CMandelbrotView::OnDraw(CDC* pDC) {
    CMandelbrotDoc* pDoc = GetDocument();
    ASSERT_VALID (pDoc);
\end{verbatim}
Insert the following code into `OnDraw` after the above two statements, then build and run the program.

```cpp
// create pens
CPen penRed (PS_SOLID, 1, RGB(255,0,0));
CPen penGreen (PS_SOLID, 1, RGB(0,255,0));
CPen penBlue (PS_SOLID, 1, RGB(0,0,255));
CPen penYellow (PS_SOLID, 1, RGB(255,255,0));
CPen penPurple (PS_SOLID, 1, RGB(255,255,0));
CPen penLightblue (PS_SOLID, 1, RGB(0,255,255));
CPen penBlack (PS_SOLID, 1, RGB(0,0,0));
CPen penWhite (PS_SOLID, 1, RGB(255,255,255));

// the values given to the variables on these lines control what is displayed and
// the scale of the display
double rcenter = -0.5;
double icenter = 0.0;
double scale = 1.5;

int by8;

double maxiterations = 96.0 / scale;
int xscale = 600;
int yscale = 350;
double iscale = scale * yscale / xscale;
double rstart = rcenter - scale;
double rstop = rcenter + scale;
double istart = icenter - iscale;
double istop = icenter + iscale;

// calculate the step factor of the drawing for the floating point variables

double rstep, istep;
if (rstart < 0 && rstop <= 0)
    rstep = ((-rstart) - (-rstop)) / ((double)xscale);
else if (0 <= rstart && 0 < rstop)
    rstep = (rstop - rstart) / ((double)xscale);
else
    rstep = ((-rstart) + rstop) / ((double)xscale);
if (istart < 0 && istop <= 0)
    istep = ((-istart) - (-istop)) / ((double)yscale);
else if (0 <= istart && 0 < istop)
    istep = (istop - istart) / ((double)yscale);
else
    istep =((-istart) + istop) / ((double)yscale);
```

// the integer step variables for the drawing coordinates
int x=0, y;

double r = rstart, i;

int color, iterations;
double oldr, newr, newi;

// the loops to calculate the Mandelbrot Set
while (x < xscale) {
    r += rstep;
    x ++;
    i = istart;
    y = 0;
pDC->MoveTo (x, y);
while (y < yscale) {
    i += istep;
    y++;
}

// calculate each X,Y point to see if it is in the Mandelbrot Set

    iterations = 0;
    newr = r;
    newi = i;
    while ( abs(newr) <= 2.0 &&
          abs(newi) <= 2.0 &&
          iterations < maxiterations) {
        iterations++;
        oldr = newr;
        newr = newr * newr - newi * newi + r;
        newi = 2 * newi * oldr + i;
    }

// select a pen based on how soon an X,Y point left the Mandelbrot Set

    if (iterations < 4) pDC->SelectObject (&penRed);
else if (iterations < 8) pDC->SelectObject (&penGreen);
else if (iterations < 16) pDC->SelectObject (&penBlue);
else {
        by8 = (iterations < 16) % 8;
        switch (by8) {
        case 0 : pDC->SelectObject(&penYellow);
            break;
        case 1 :pDC->SelectObject(&penPurple);
            break;
        case 2 :pDC->SelectObject(&penLightblue);
break;
case 3 : pDC->SelectObject(&penBlack);
break;
case 4 : pDC->SelectObject(&penWhite);
break;
case 5 : pDC->SelectObject(&penRed);
break;
case 6 : pDC->SelectObject(&penGreen);
break;
default : pDC->SelectObject(&penBlue);
}
}

// draw the point in the current color

pDC->LineTo(x, y);

}