

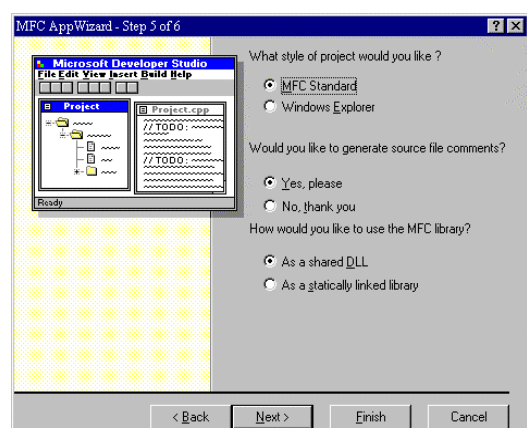
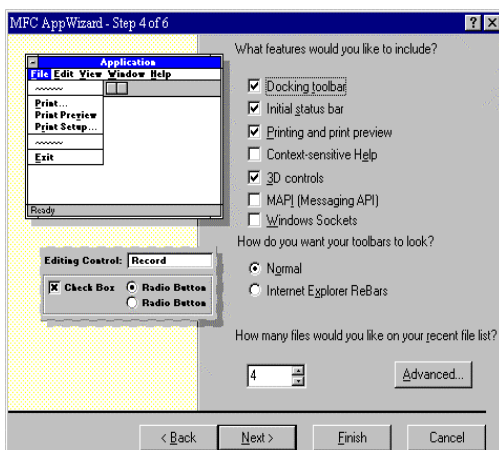
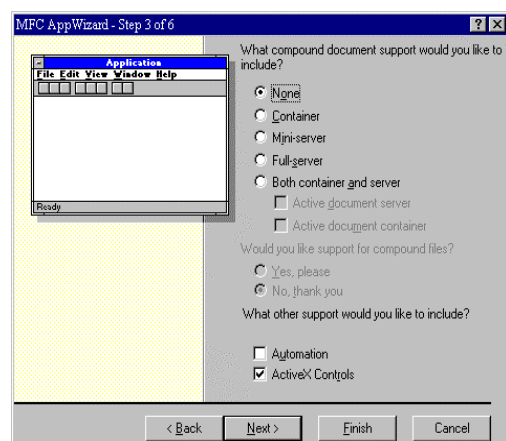
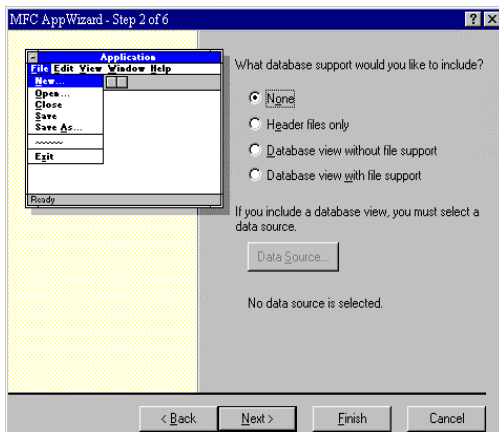
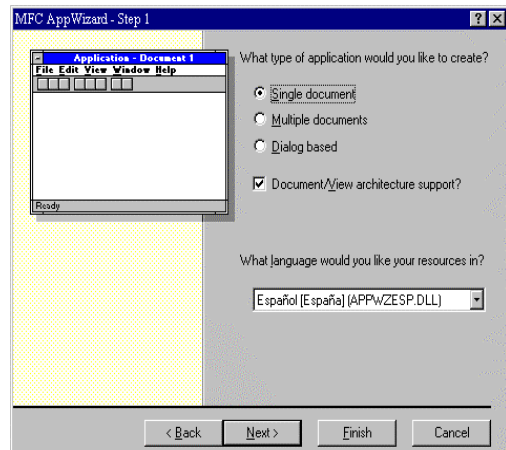
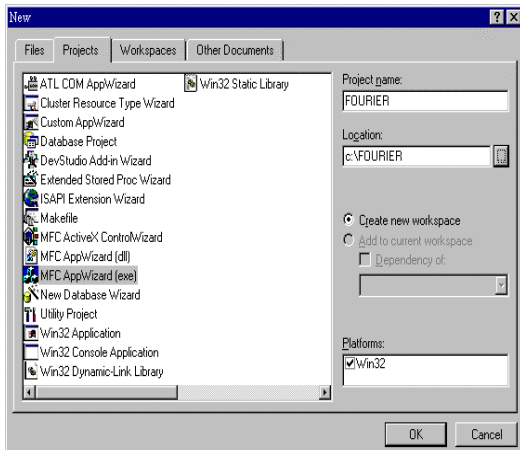
LABORATORIO N°5**OBJETIVO: SERIES DE FOURIER**

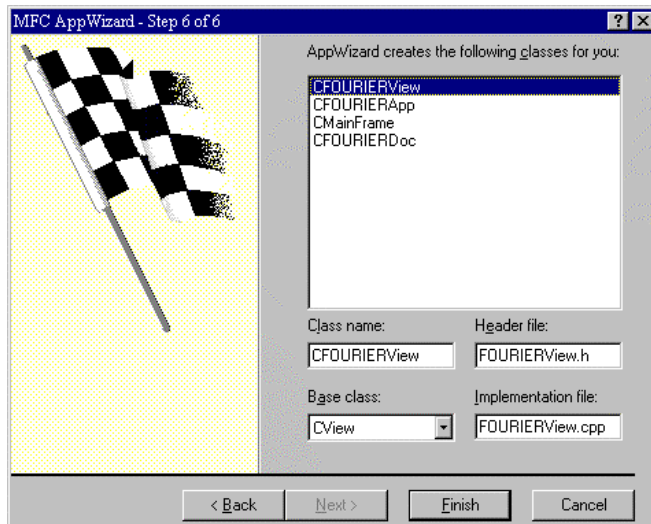
Utilizar las clases MFC: CBrush, CPen, CPaintDC y el contexto de dispositivo(device context -DC), para las aplicaciones con Grafico y Color.

Utilizar un **sistema de coordenadas** que se pueda ajustar a la escala que se desea.

PARTE I

PASO 1: Seleccione **File/New** . Siga las opciones que se muestran.

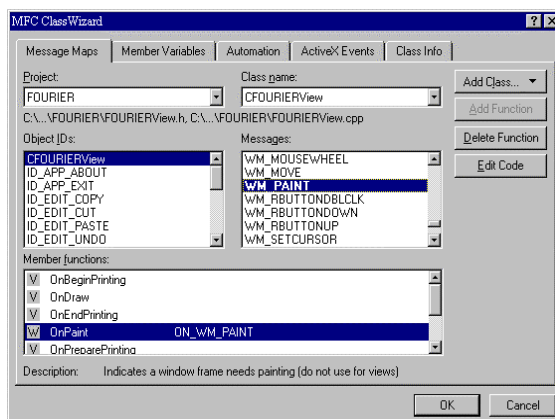




Presione Finish y continúe .

PARTE II: INTRODUCIENDO CODIGO

PASO 1:



En esta aplicación, se incorporará al código del programa una función miembro **OnPaint ()** para procesar los mensajes **WM_PAINT**.

PASO 2. Escriba el código para la función **OnPaint()**

```
void CFourierView::OnPaint()
{
```

```
    CPaintDC dc(this); // device context for painting
```

```
    static DWORD dwColor [9] = { RGB (0,0,0),           // negro
                                RGB (245,0,0),           // rojo
                                RGB (0,245,0),           // verde
                                RGB (0,0,245),           // azul
                                RGB(245,245,0),          // amarillo
                                RGB (245,0,245),         // magenta
                                RGB(0,245,245),          // cyan
                                RGB (127,127,127),       // gris
                                RGB (245,245,245) };     // blanco
```

```
    int i, j, ltitulo, ang;
    double y, yp;
    CBrush n_brocha;
    CBrush *v_brocha;
    CPen n_pincel;
    CPen *v_pincel;
```

```
    // Crea una superficie de dibujo personalizada
    dc.SetMapMode(MM_ISOTROPIC); //Establecer la escala del sistema de coordenadas
```

```

dc.SetWindowExt(500,500); //Tamaño de la ventana
dc.SetViewportExt(m_cxClient, -m_cyClient); //tamaño del area de dibujo (ventana grafica)
dc.SetViewportOrg(m_cxClient/20, m_cyClient/2); //establecer el origen de coordenadas

ang=0;
yp=0.0;

n_pincel.CreatePen(BS_SOLID, 2, RGB (0, 0, 0) );
v_pincel = dc.SelectObject (&n_pincel);

//dibuja los ejes de coordenadas x & y
dc. MoveTo (0,240);
dc.LineTo (0,-240);
dc.MoveTo(0,0);
dc.LineTo(400,0);
dc.MoveTo(0,0);
//dibuja el espectro de onda actual de Fourier
for (i=0; i<=400; i++) {
    for (j=1; j<=nterms; j++) {
        y=(150.0/ ((2.0*j)-1.0))*sin(((j*2.0)-1.0)*0.015708*ang);
        yp=yp+y;
    }
    dc.LineTo (i, (int) yp);
    yp-=yp;
    ang++;
}
// preparacion para rellenar el interior del espectro.
n_brocha.CreateSolidBrush (dwColor [7] );
v_brocha=dc.SelectObject(&n_brocha);
dc.ExtFloodFill (150, 10, dwColor [0], FLOODFILLBORDER);
dc.ExtFloodFill (300, -10, dwColor [0], FLOODFILLBORDER);
// imprime el titulo del espectro
ltitulo = strlen (mititulo);
dc.TextOut(200-(ltitulo*8/2), 185,mititulo, ltitulo);

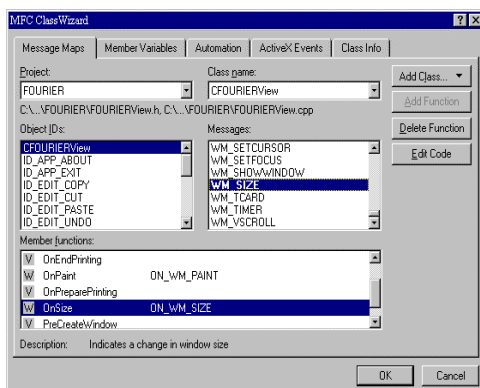
// borra los objetos brocha
dc.SelectObject(v_brocha);
n_brocha.DeleteObject( );

}

```

PASO 3:

Determinación del tamaño actual de la ventana



La función miembro **OnSize()**
Devuelve el tamaño actual de la ventana cliente.

Siempre que se modifica el tamaño de la ventana se genera un mensaje **WM_SIZE**

```

void CFOURIERView::OnSize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);

    m_cxClient=cx,
    m_cyClient=cy;
}

```

}

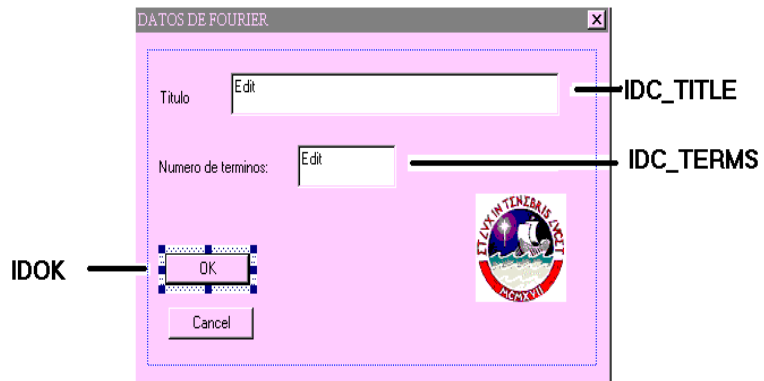
PASO 4: Declare las siguientes variables:

```
FOURIERView.cpp
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
#include "math.h"
int m_cxClient, m_cyClient;
int nterms=500;
char mititulo[80]="500 ARMONICOS DE FOURIER";

// FOURIERView.cpp
```

PASO 5:

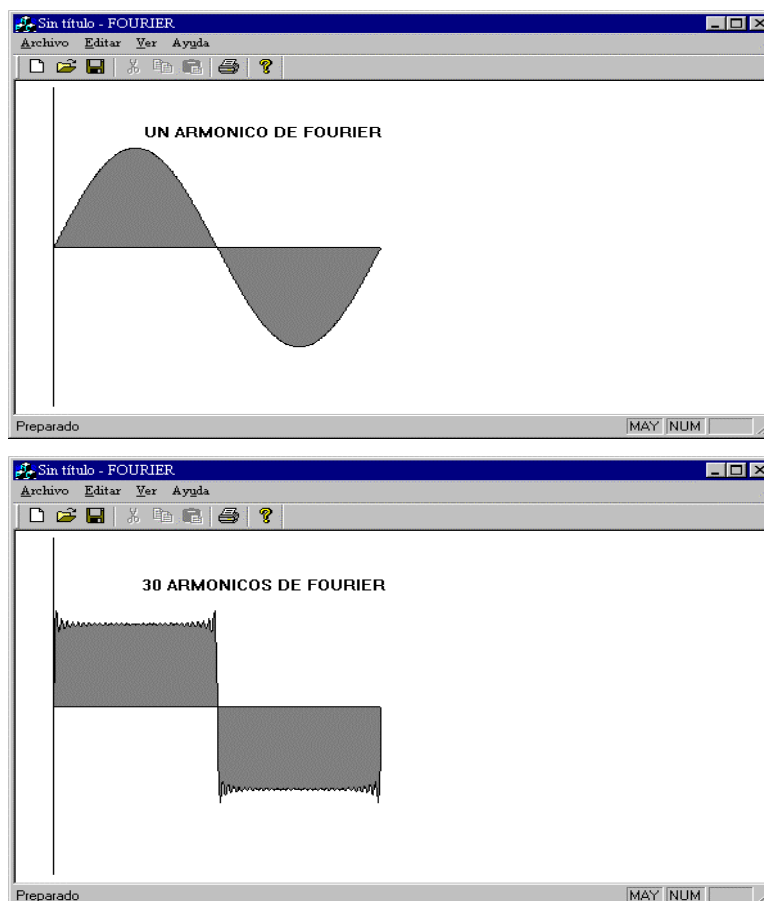
```
void CFourierView::OnFileNew()
{
    CDatosDialog dlg;
    dlg.DoModal();
}
```

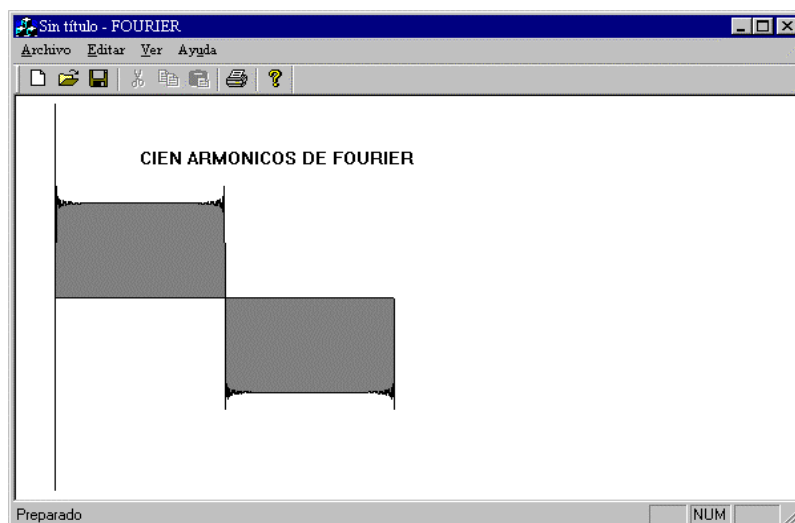
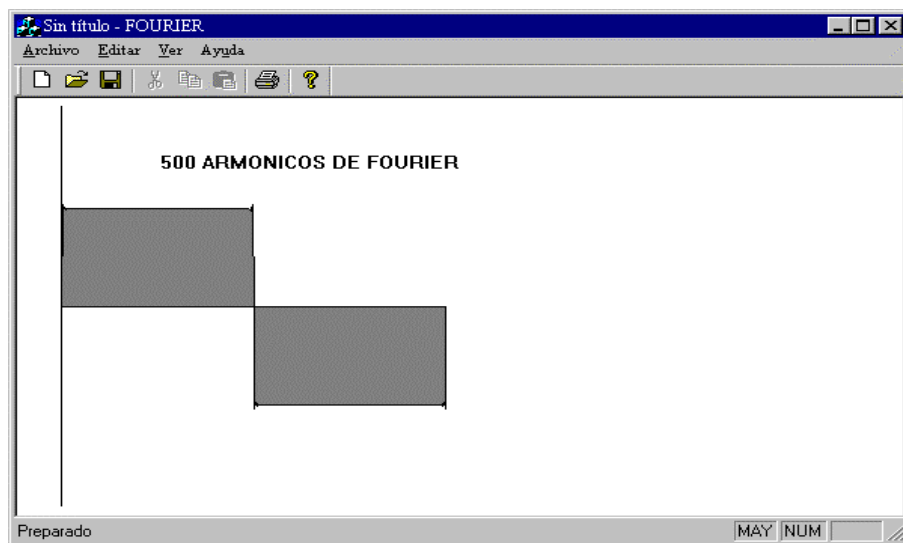


```
void CDatosDialog::OnOK()
{
    GetDlgItemText(IDC_TITLE, mititulo, 80);
    nterms = GetDlgItemInt(IDC_TERMS, NULL, 0);
    CDialog::OnOK();
}
```

PARTE III :

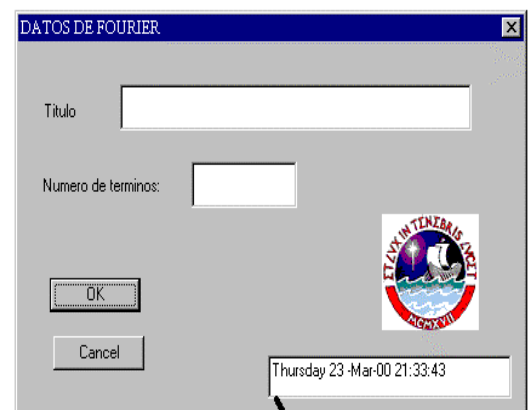
Resultados del PROYECTO **FOURIER**





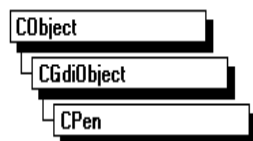
PARTE IV: Codigo para la Fecha Y Hora

```
void CDatosDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CDatosDialog)
    CTime FechaHora = CTime::GetCurrentTime();
    CString FechaHoraTxt = FechaHora.Format("%A %d -%b-%y
    %X");
    SetDlgItemText(IDC_TIME, FechaHoraTxt);
   //}}AFX_DATA_MAP
}
```



IDC_TIME

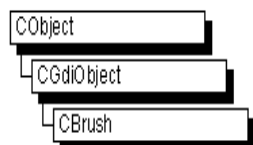
OBSERVACIONES

CPen

La clase **CPen** permite la pluma a utilizar con las funciones de dibujo. Esta herramienta afecta a la forma en la que son dibujadas las líneas: líneas sólidas, líneas discontinuas, etc.

Así como a los bordes de las figuras cerradas: Rectángulos, elipses, etc.

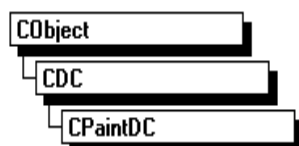
Cuando se crea un contexto de dispositivo, existe una pluma por defecto que dibuja líneas negras sólidas que tienen un ancho de un pixel.

CBrush

La Clase **CBrush** permite definir el pincel a utilizar con las funciones de dibujo.

Esta herramienta afecta a la forma en la que es pintado el interior de las figuras cerradas.

Cuando se crea un contexto de dispositivo, existe un pincel por defecto que pinta de blanco el interior de las figuras cerradas.

CPaintDC

La clase **CPaintDC** se ha diseñado para utilizar con la función **OnPaint**. Para visualizar texto o gráficos en el área de trabajo de una ventana en respuesta a un mensaje **WM_PAINT**.

Hay que crear un objeto **DC** perteneciente a la clase **CPaintDC** o una clase derivada de ella.

Utilice un objeto **CPaintDC** solamente con una función destinada a manipular el mensaje **WM_PAINT**.

VISUALIZAR LA FECHA Y HORA

La clase **CTime** permite obtener información relativa a la fecha y la hora actuales.

FORMATO	DESCRIPCION
%a	nombre del día de la semana abreviado
%A	nombre del día de la semana completo
%W	día de la semana (0 a 6; domingo = 0)
%b	nombre del mes abreviado
%B	nombre del mes completo
%d	día del mes (1 a 31)
%d	mes (1 a 12)
%y	año (0 a 99)
%Y	año (cuatro dígitos)
%H	hora (0 a 24)
%I	hora (1 a 12)
%p	AM/PM
%M	minutos (0 a 59)
%S	segundos (0 a 59)
%x	fecha (mm/dd/aa)
%X	hora (hh:mm:ss)

LINEAS RECTAS:

BOOL LineTo(int x, int y);

BOOL LineTo(POINT point);

LineTo:

Pinta una línea desde la posición actual de la pluma hasta el punto lógico especificado por los argumentos **x** e **y** o por argumento punto.

Por defecto, la posición inicial de la pluma es el punto lógico(0,0).Puede establecer otro punto utilizando la función **MoveTo**.

ESCALAS VARIABLES:

- Las escalas **MM_ISOTROPIC** y **MM_ANISOTROPIC**, son variables, esto es, cuando se utiliza una de estas escalas es posible cambiar tanto el origen de coordenadas como el factor de escala.
- La escala **MM_ISOTROPIC** expresa las coordenadas lógicas **x** e **y** en unidades lógicas de tamaño físico idéntico, lo que permite pintar círculos y cuadrados perfectos.

CDC:SetMapMode

Permite ajustar el sistema de coordenadas a una de las escalas estándar.

CDC:SetWindowExt

Permite establecer las medidas de la ventana asociada con el contexto de dispositivo.

CDC:SetViewportExt

Permite establecer las medidas de la ventana gráfica(área de dibujo).

CDC:SetViewportOrg

Tiene dos parámetros que especifican las coordenadas (x, y) en “unidades de dispositivo” del nuevo origen de coordenadas.