APENDICE B

MODELO DE CLASES DEL SISTEMA

B. 1 Descripción del proyecto

Diseño O-O del sistema Reconocedor de Partituras Musicales, empleando la simbología de Coud y Yourdon.

B.2 Diagramas de clases UML

B.2.1 Componentes

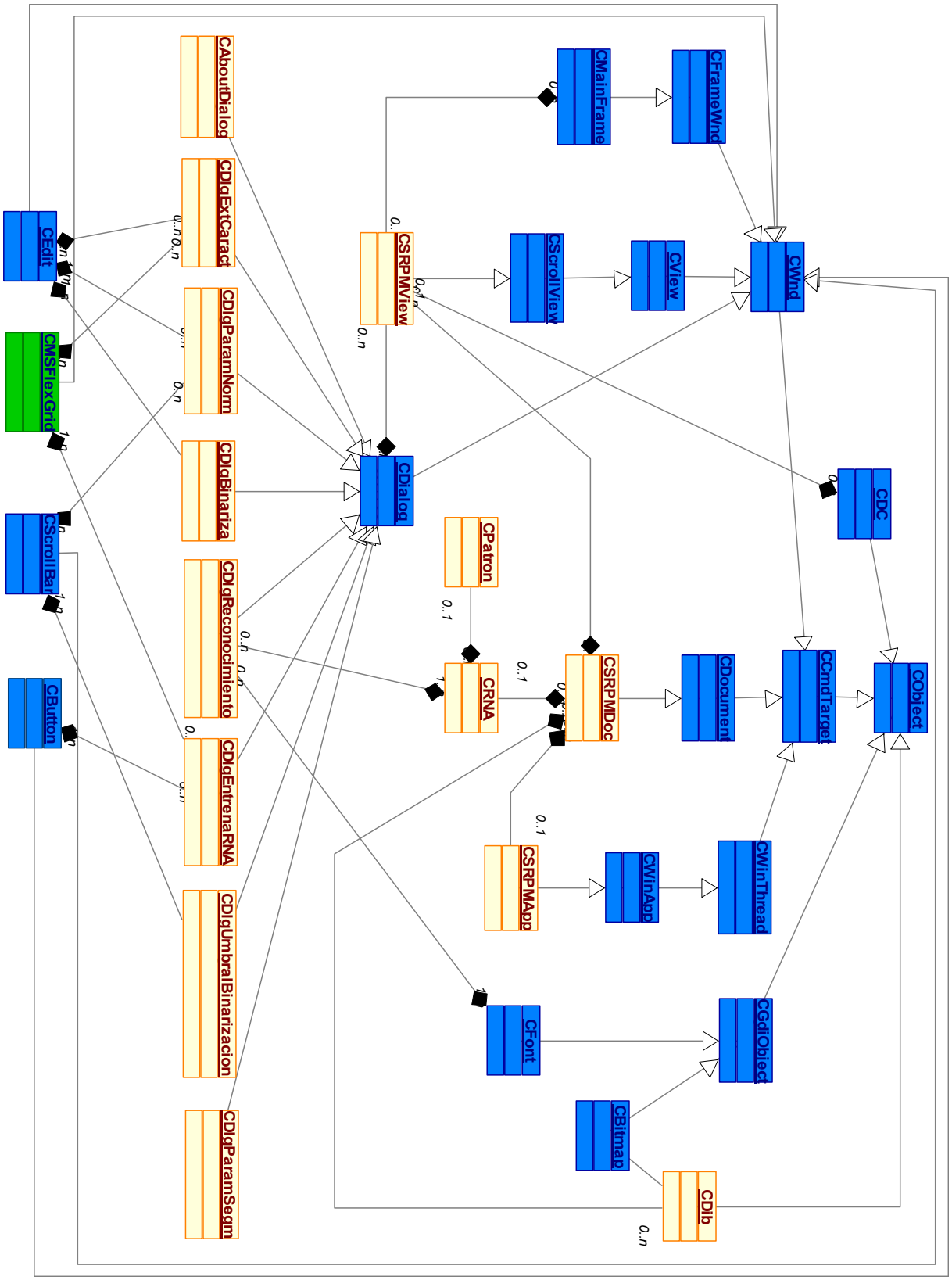
B.2.1.1 Paquetes (Clases azules en diseño O-O)

- *Microsoft Foundation Class*

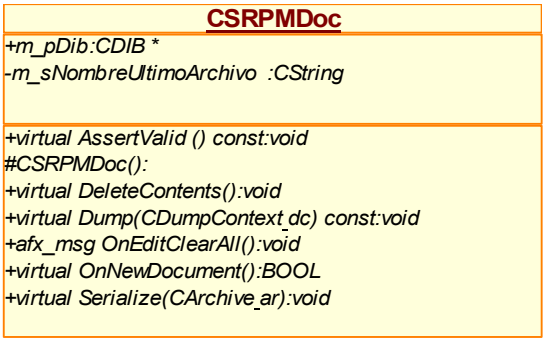
Descripción:

El esqueleto de trabajo de la MFC es un elemento poderoso que permite construir aplicaciones basadas en el trabajo de expertos programadores en Windows. La MFC reduce el tiempo de desarrollo; hace el código más portable; provee un tremendo soporte sin reducir la libertad y flexibilidad en la programación; y da un acceso fácil a la "programación difícil" de los elementos y tecnologías empleadas en interfases a usuarios tales como la tecnología Active, OLE, y programación de Internet. Incluso MFC simplifica la programación de base de datos a través de los objetos de acceso a datos (DAO) y la conectividad de apertura de base de datos (ODBC)

- ***Título:*** Diseño Orientado a Objetos



B.2.1.2 Clases (Clases de color naranja en diseño O-O)



CSRPMDoc

Descripción: Esta clase es la encargada de la creación y manipulación del objeto documento asociado con la aplicación S.R.P.M. Entre sus funciones principales está la de asociar a las partituras como objetos CDIB con el objeto documento de SRPM.

Métodos:

- *AssertValid()const* *Tipo devuelto: void*

Documentación: Ejecuta una prueba de validación al objeto documento al verificar su estado interno. AssertValid() puede intentar validar al objeto documento y entonces terminar la ejecución del programa un mensaje en donde se enliste el número de línea en donde ésta validación falló.

- *virtual DeleteContents()* *Tipo devuelto: void*

Documentación: Función polimórfica cuya definición se encuentra en la clase CDocument y que en CSRPMView es la encargada de borrar la dirección del dato miembro apuntador m_pDib para después hacer que éste apunte a un nuevo CDib asociado al documento.

- *virtual Dump(CDumpContext& dc) const Tipo devuelto: void*

Documentación: Función polimórfica cuya definición se encuentra en la clase CObject y que no tiene una redefinición de su funcionamiento por default en CSRPMDoc.

- *afx_msg OnEditClearAll() Tipo devuelto: void*

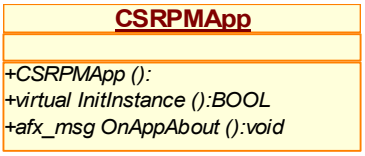
Documentación: Función miembro llamada por el esqueleto de trabajo cuando el usuario selecciona la opción de "borrar todo" en el menú de Edición de S.R.P.M. Su misión es la de llamar a las funciones virtuales correspondientes para eliminar el Dib que se encuentra actualmente cargado desasociando el apuntador m_pDib a la dirección del mencionado Dib y asociándolo a la dirección de la nueva partitura en éste momento en blanco. Después se llamará a la función miembro de CDocument SetModifiedFlag la cual se ejecuta después de haber realizado cualquier modificación al documento (ver definición en CDocument) y finalmente llama a la función miembro de CDocument UpdateAllViews cuya definición se encuentra en dicha clase y que actualiza de alguna manera todas las vistas asociadas al documento que ha recibido la modificación.

- *virtual OnNewDocument() Tipo devuelto: BOOL*

Documentación: Función polimórfica definida en CDocument y que inicializa un nuevo documento. Es llamada por el esqueleto de trabajo como parte de la serie de actividades que deben ser ejecutadas como parte de un comando File New.

- *virtual Serialize(CArchive& ar) Tipo devuelto: void*

Documentación: Lee o escribe el objeto CDIB desde o hacia un archivo.



CSRPMApp

Descripción: Esta clase es derivada de CWinApp y contiene las funciones miembro para inicializar el sistema S.R.P.M.como una aplicación de Windows, así como proveer estos elementos para que dicha aplicación corra. Contiene a un único

objeto derivado de CWinApp llamado “theApp”. Este objeto es construido cuando otros objetos globales de C++ son construidos y está listo y disponible cuando Windows hace un llamado a la función WinMain la cual es provista por MFC. La declaración del objeto theApp es a nivel global.

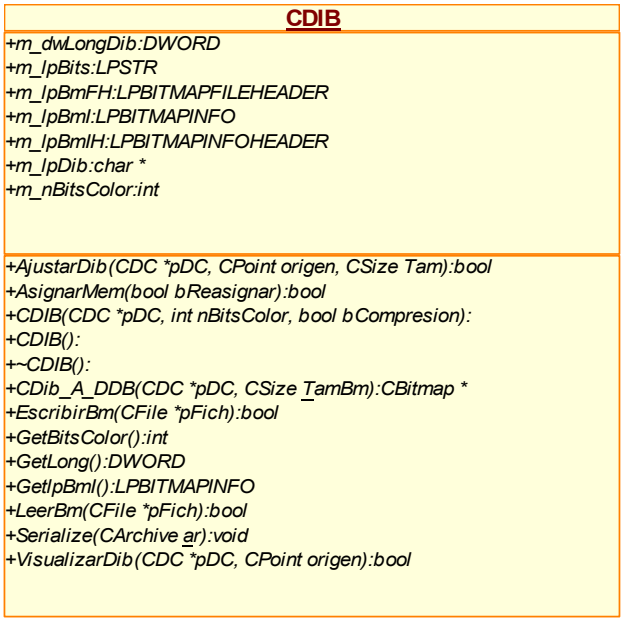
Métodos:

- *virtual InitInstance ()* *Tipo devuelto: BOOL*

  Documentación: Función polimórfica que permite sobrescribir la manera en la que se debe inicializar cada nueva instancia de la aplicación S.R.P.M., cuando corre bajo Windows, ya que Windows permite muchas copias del mismo programa correr al mismo tiempo. La inicialización de una aplicación se divide conceptualmente en dos fases: la inicialización de la aplicación cuando esta corre por primera vez y la inicialización del resto de las copias de S.R.P.M. cuando corren bajo Windows.

- *afx_msg OnAppAbout ()* *Tipo devuelto: void*

  Documentación: Función miembro que es llamada por el esqueleto de trabajo de la aplicación cuando el usuario selecciona la opción “Acerca de…” del menú principal y hace que se despliegue la caja de diálogo correspondiente a los derechos de autor de S.R.P.M.



CDIB

Descripción: Se dice que un mapa de bits es independiente del dispositivo por que contiene una tabla de color. Un mapa de bits independiente de dispositivo (DIB) no es un objeto GDI. Lo que hace un la GDI es obtener el DIB y convertirlo en un DDB compatible con algún dispositivo de salida; en este caso se pierde información del color del DIB. El propósito principal de los DIB es permitir intercambiar mapas de bits entre programas.

Un objeto DIB utiliza un formato que permite varias resoluciones de color. En contraste con un objeto DDB, un DIB utiliza un formato externo y cuando se carga, aparece en el sistema igual que un objeto que hubiera sido creado por una aplicación utilizando CreateBitmap, CreateCompatibleBitmap, CreateBitmapIndirect o CreateDIBitamp.

Un DIB consta de dos partes: una cabecera que describe el formato de los bits, y los bits del mapa de bits (el array de bits comienza por la línea inferior de los píxeles en lugar de la superior). La cabecera contiene el formato de color, una tabla de colores y el tamaño del mapa de bits. La estructura actual de un DIB soporta seis profundidades o resoluciones de color: 1 bit (blanco y negro), 4 bits (16 colores), 8 bits (256 colores) , 16 bits (65536 colores), 24 bits (16 millones de colores) y 32 bits. En los DIB de resolución 1 bit, 4 bits y 8 bits, los píxeles están definidos por índices de una tabla de colores. En la resolución de 24 bits cada píxel está descrito por un valor de 24 bits (valor RGB), 1 byte para cada uno de los colores rojo, verde y azul.

La extensión por omisión de un archivo que almacena un DIB es BMP. Un archivo de este tipo consta de una estructura BITMAPFILEHEADER seguido de otra estructura BITMAPINFOHEADER, de la tabla de colores y del propio mapa de bits.

Métodos:

- *AjustarDib(CDC *pDC, CPoint origen, CSize Tam) Tipo devuelto: bool*

Documentación: Envía el objeto CDIB a la pantalla o a la impresora sin necesidad de crear un DDB y ajustándolo a un rectángulo determinado.

- *AsignarMem(bool bReasignar)* *Tipo devuelto: bool*

Documentación: Permite asignar memoria para un DIB. Para ello, la función miembro AsignarMem utiliza la macro GlobalReAllocPtr para reasignar memoria para un DIB existente o la macro GlobalAllocPtr para asignar memoria para un DIB nuevo. Estas macros están definidas en Windows.h

- *CDib_A_DDB(CDC *pDC, CSize &TamBm)* *Tipo devuelto: CBitmap **

Documentación: Crea un mapa de bits dependiente de dispositivo a partir de un objeto CDIB. La función devuelve el mapa de bits que había seleccionado originalmente en el contexto de dispositivo y en su parámetro TamBm el tamaño del mapa de bits creado. Para realizar esta operación se utiliza la función CreateDIBitamp. Antes de llamar a CreateDIBitamap hay que selccionar en el contexto de dispositivo referenciado por el parámetro pDC un mapa de bits de las mismas características que el que deseamos crear. De no hacerlo así, CreateDIBitmap generará un mapa de bits monocromo.

- *EscribirBm(CFile *pFich)* *Tipo devuelto: bool*

Documentación: Esta función escribe un DIB de un objeto CDib en un archivo. El archivo debe haber sido abierto o creado con éxito.

- *GetBitsColor()* *Tipo devuelto: int*

Documentación: Función que sirve para acceder al número de bits por píxel.

- *GetLong()* *Tipo devuelto: DWORD*

Documentación: Función que sirve para obtener el número total de bytes del archivo que contiene el mapa de bits.

- *GetlpBmI()* *Tipo devuelto: LPBITMAPINFO*

Documentación: Esta función se emplea para obtener la dirección de la estructura BITMAPINFO que contiene información del DIB y de la tabla de colores.

- *LeerBm(CFile *pFich)* *Tipo devuelto: bool*

Documentación: No Esta función lee un DIB de un archivo en el objeto CDib. El archivo debe haber sido abierto con éxito. Si el archivo es un BMP, la lectura comienza desde el principio del archivo. Si es un documento, comienza desde puntero de archivo actual.

- *Serialize(CArchive &ar)* *Tipo devuelto: void*

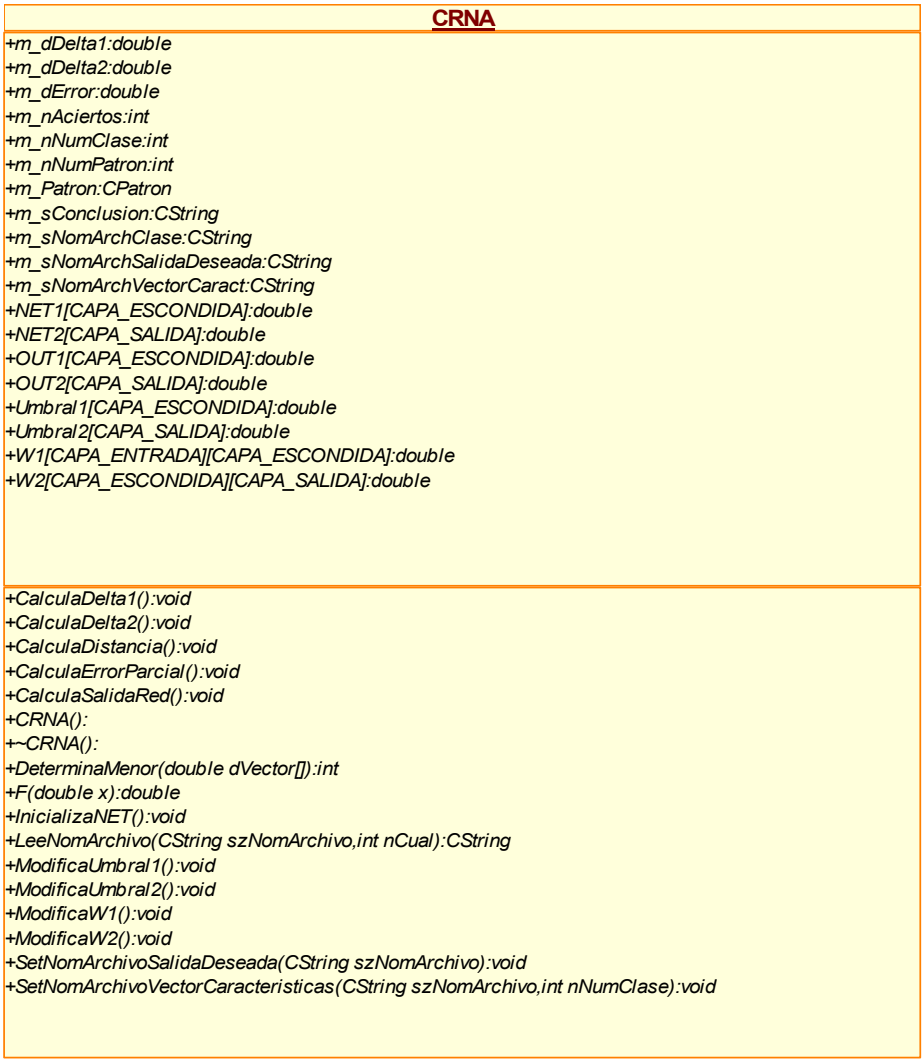
Documentación: La función CDib::Serialize, que sobrescribe a la función CObject::Serialize de la MFC, llama a las funciones miembro LeerBm y EscribirBm.

- *VisualizarDib(CDC *pDC, CPoint origen)* *Tipo devuelto: bool*

Documentación: Esta función muestra el objeto CDib en la visualización con una llamada a la función StretchDIBits de Win32. El mapa de bits se ampliará lo necesario para encajar en el rectángulo especificado.

CRNA

Descripción: La clase RNA sirve para definir objetos redes neuronales, de hecho S.R.P.M. maneja por composición dos objetos RNA uno para definir y manipular la red de tonos y otro para la red de notas musicales. Es una clase que permite entrenar redes neuronales mediante el algoritmo de retropropagación, los objetos RNA estarán definidos por tres parámetros importantes: COEFICIENTE_APRENDIZAJE, NUM_CLASES y TAM_CLASE y que varian de red a red. Esta clase establece una relación de composición con la clase CPatrón en donde se definen las características escenciales de cada patrón empleado para entrenar a una red.



Métodos:

- *CalculaDelta1()* *Tipo devuelto: void*

Documentación: Función miembro que calcula los valores del vector delta1 y que son los valores asociados a cada neurón en la capa escondida con los cuales se efectuará posteriormente la modificación de pesos y umbrales de dicha capa neuronal.

- *CalculaDelta2()* *Tipo devuelto: void*

Documentación: Función miembro que calcula los valores del vector delta2 y que son los valores asociados a cada neurón en la capa de salida con los cuales se efectuará posteriormente la modificación de pesos y umbrales de dicha capa neuronal durante la retropropagación.

- *CalculaDistancia()* *Tipo devuelto: void*

Documentación: Función miembro que efectúa los cálculos necesarios para determinar todas las distancias espaciales existentes en un momento dado entre la salida obtenida por la red y las salidas deseadas de cada uno de las clases que conforman el conjunto característico con el que se está entrenando la red.

- *CalculaErrorParcial()* *Tipo devuelto: void*

Documentación: Función miembro que calcula el error parcial de cada patrón para posteriormente acumularlo en el error total por barrida. Este cálculo implica la evaluación del error medio al cuadrado entre la salida deseada del patrón en turno y la salida obtenida por aplicar los pasos de la red.

- *CalculaSalidaRed()* *Tipo devuelto: void*

Documentación: Ejecuta los pasos necesarios para determinar el valor de la salida obtenida de la red mediante un cálculo tradicional para redes feedforward. Este cálculo implica la multiplicación vectorial de la entrada a cada capa neuronal la cual puede ser una entrada del exterior (vector de características) o la salida de una capa neuronal previa (OUT1) haciendo el producto punto con la matriz de pesos de la capa neuronal en turno menos el valor umbral definido para dicha capa neural. Ese valor es pasado por una función filtro llamada función de activación.

- *DeterminaMenor(double dVector[])* *Tipo devuelto: int*

Documentación: Esta función miembro determina cual de las distancias calculadas por la función miembro *CalculaDistancia()*, es la menor ya que eso indicará la clase con la cual la red estará asociando al patrón previamente alimentado.

- *F(double x)* *Tipo devuelto: double*

Documentación: Función miembro que aplica a las salidas de cada capa neuronal una función filtro conocida como función de activación, en el caso de S.R.P.M. esta función es la función sigmoide.

- *InicializaNET()* *Tipo devuelto: void*

Documentación: Función miembro que inicializa los valores de los vectores NET a ceros.

- *LeeNomArchivo(CString szNomArchivo,int nCual) Tipo devuelto: CString*

Documentación: Función miembro que es llamada cada vez que se requiere leer el nombre de un archivo que se encuentra escrito en el archivo szNomArchivo. El parámetro nCual determina cual de todos los nombres escritos en el archivo szNomArchivo es el que deberá leerse.

- *ModificaUmbral1()* *Tipo devuelto: void*

Documentación: Esta función es llamada cada vez que se está aplicando la retropropagación una vez que ya han sido calculados los valores del vector delta1 y su acción es la de modificar los valores actuales de los umbrales asociados con cada uno de los neuronas en la capa de escondida.

- *ModificaUmbral2()* *Tipo devuelto: void*

Documentación: Esta función es llamada cada vez que se está aplicando la retropropagación una vez que ya han sido calculados los valores del vector delta2 y su acción es la de modificar los valores actuales de los umbrales asociados con cada uno de los neuronas en la capa de salida.

- *ModificaW1()* *Tipo devuelto: void*

Documentación: Esta función es llamada cada vez que se está aplicando la retropropagación una vez que ya han sido calculados los valores del vector delta1 y su acción es la de modificar los valores actuales de los pesos asociados con cada una de las conexiones sinápticas que unen a los neurones de la capa de entrada con los neuronas de la capa escondida.

- *ModificaW2()* *Tipo devuelto: void*

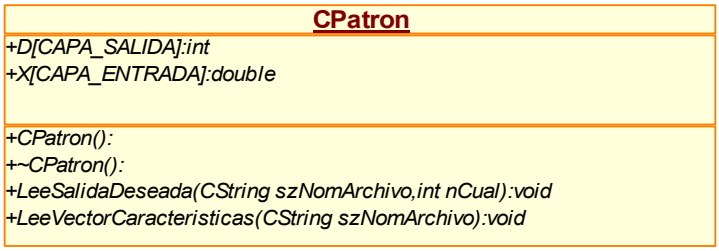
Documentación: Esta función es llamada cada vez que se está aplicando la retropropagación una vez que ya han sido calculados los valores del vector delta2 y su acción es la de modificar los valores actuales de los pesos asociados con cada una de las conexiones sinápticas que unen a los neurones de la capa escondida con los neuronas de la capa de salida.

- *SetNomArchivoSalidaDeseada(CString szNomArchivo) Tipo devuelto: void*

Documentación: Su función es muy similar a la de *LeeNomArchivo(CString szNomArchivo,int nCual)* solamente que se emplea para leer el nombre del archivo que contiene los datos de las salidas deseadas de cada una de las clases con las que se entrena a la red neuronal.

- *SetNomArchivoVectorCaracteristicas(CString szNomArchivo,int nNumClase) Tipo devuelto: void*

Documentación: Su función es muy similar a la de *LeeNomArchivo(CString szNomArchivo,int nCual)* solamente que se emplea para leer el nombre del archivo que contiene los datos de los vectores de características de cada uno de los patrones que están siendo alimentados a la red neuronal.



CPatron

Descripción: La clase CPatron es una clase que da la funcionalidad necesaria para crear y manipular los patrones que serán alimentados a la red neuronal vía un proceso de entrenamiento o de reconocimiento.

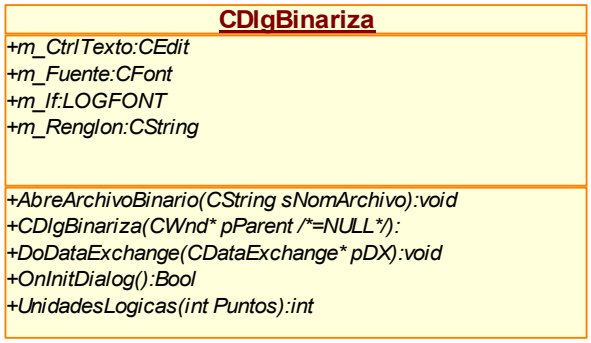
Métodos:

- *LeeSalidaDeseada(CString szNomArchivo,int nCual) Tipo devuelto: void*

Documentación: Esta función es llamada siempre que se necesita leer de un archivo los valores que se guardarán en el vector D (salida deseada) del objeto CPatron.

- *LeeVectorCaracteristicas(CString szNomArchivo) Tipo devuelto: void*

Documentación: Esta función sirve para obtener de un archivo los valores que se guardarán en el vector X (vector de características) del objeto CPatron.



CDlgBinariza

Descripción: La clase CDlgBinariza es la clase que maneja la caja de diálogo "Partitura Binarizada", mediante la cual se muestra al usuario el resultado de aplicar a una nota musical el proceso de Binarización. CDlgBinariza muestra el contenido de un archivo de texto resultante de aplicar una llamada a la función miembro OnOpcionesBinarizacion() de la clase CSRPMView.

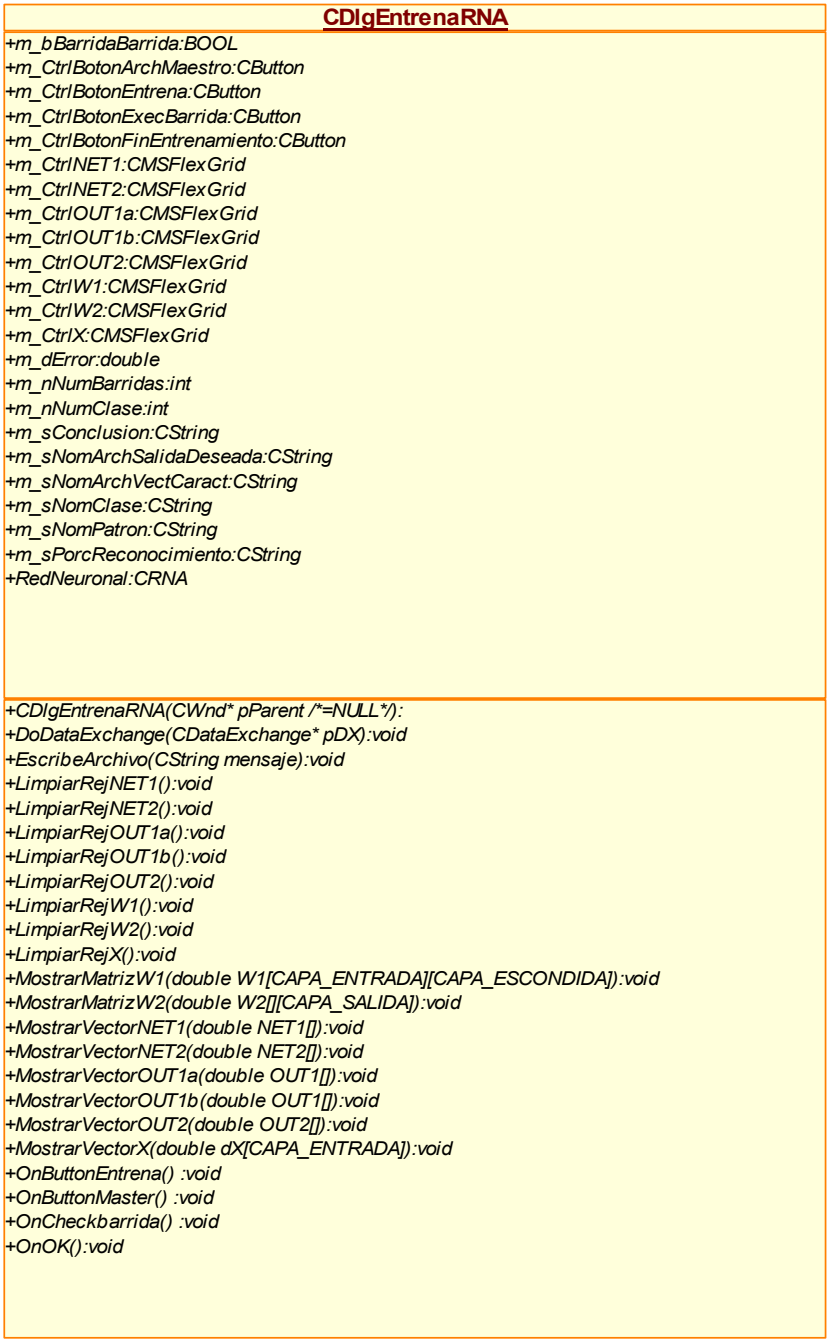
Métodos:

- *AbreArchivoBinario(CString sNomArchivo) Tipo devuelto: void*

  Documentación: Esta función es llamada cuando se requiere instanciar el dato miembro m_Renglon con el texto contenido en el archivo producido como resultado de una llamada a la función OnOpcionesBinarización() de la clase CSRPMView. El parámetro *sNomArchivo* es proporcionado por el dato miembro m_sNombreUltimoArchivo de la clase CSRPMDoc.

- *DoDataExchange(CDataExchange* pDX) Tipo devuelto: void*

  Documentación: Inmediatamente después de que la caja de diálogo "Partitura Binarizada" es desplegada, el esqueleto de trabajo transfiere por medio de un mecanismo DDX (Dialog Data eXchange) los valores de los datos miembro a los controles en la caja de diálogo donde aparecerán cuando la caja de diálogo por sí misma aparezca como respuesta a una llamada DoModal o Create.



CDlgEntrenaRNA

Descripción: La clase CDlgEntrenaRNA es la clase que maneja a la caja de diálogo "Entrenamiento Red Neuronal". Esta es la caja de diálogo que sirve como interface de entrada y salida al usuario cuando se realiza el entrenamiento de la red neuronal. La clase CDlgEntrenaRNA utiliza ésta caja de diálogo para recibir por parte del usuario el nombre del archivo maestro con el que se iniciará el proceso de entrenamiento. El usuario tiene además la opción de determinar si desea hacer una ejecución del algoritmo de entrenamiento de retropropagación barrida a barrida o si sólo recibirá resultados hasta el término del entrenamiento.

La caja de diálogo "Entrenamiento Red Neuronal" tiene dos secciones que sirven como interface de salida.

Información General del Entrenamiento, donde se le proporciona al usuario información como: el número y nombre de clase que está siendo alimentada a la red en un momento dado el nombre del patrón perteneciente a esa clase con el que se está trabajando el número de barridas que se llevan ejecutadas en ese instante el error total por barrida acumulado y la conclusión a la que se llegó una vez que el patrón se le ha terminado su procesamiento.

Red Neuronal, donde se le proporciona al usuario información muy específica sobre los resultados de los cálculos que se están llevando a cabo en un momento dado. Dichos resultados incluyen el vector de características del patrón en turno, las matrices de pesos que representan la intensidad de las conexiones sinápticas de todas las capas de la red y los valores de salida de cada capa neuronal.

Al término de la ejecución del entrenamiento CDlgEntrenaRNA emitirá un aviso indicando que la condición de fin de entrenamiento se ha alcanzado y activando el botón de comando "FIN ENTRENAMIENTO". Acto seguido CDlgEntrenaRNA solicitará al usuario proporcione un nombre al archivo donde se guardarán los pesos y umbrales finales que constituyen la configuración "ideal" de la red para reconocer patrones en un futuro.

Métodos:

- *EscribeArchivo(CString mensaje)* *Tipo devuelto: void*

  Documentación: Función que escribe en un archivo de texto información importante pertinente al proceso de entrenamiento de una red barrida a barrida, como es, el número de barrida, el error total en esa barrida y el porcentaje de reconocimiento alcanzado por la red en dicha barrida.

- *LimpiarRejNET1()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores escritos en la rejilla NET1.

- *LimpiarRejNET2()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores escritos en la rejilla NET2.

- *LimpiarRejOUT1a()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores escritos en la rejilla OUT1 parte superior.

- *LimpiarRejOUT1b()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores escritos en la rejilla OUT1 parte inferior

- *LimpiarRejOUT2()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores escritos en la rejilla OUT2.

- *LimpiarRejW1()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores de los pesos escritos en la rejilla W1.

- *LimpiarRejW2()* *Tipo devuelto:* void

  Documentación: Esta función limpia los valores de los pesos escritos en la rejilla W2.

- *LimpiarRejX()* *Tipo devuelto: void*

  Documentación: Esta función limpia los valores del vector de características escritos en la rejilla X.

- *MostrarMatrizW1(double W1[CAPA_ENTRADA][CAPA_ESCONDIDA])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de los pesos correspondientes a las conexiones sinápticas que conectan a los neurones de la capa de entrada con los neurones de la capa escondida, en la rejilla W1 mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarMatrizW2(double W2[][CAPA_SALIDA])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de los pesos correspondientes a las conexiones sinápticas que conectan a los neurones de la capa escondida con los neurones de la capa de salida, en la rejilla W2, mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarVectorNET1(double NET1[])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de las salidas parciales de los neurones de la capa escondida, antes de ser filtrados por la función de activación, en la rejilla NET1 mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarVectorNET2(double NET2[])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de las salidas parciales de los neurones de la capa de salida, antes de ser filtrados por la función de activación, en

la rejilla NET2 mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarVectorOUT1a(double OUT1[])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de las salidas obtenidas de los neurones de la capa escondida, y que corresponden a los pulsos nerviosos producidos por esa capa neuronal. Se muestran en la rejilla OUT1 parte superior mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarVectorOUT1b(double OUT1[])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de las salidas obtenidas de los neurones de la capa escondida, y que corresponden a los pulsos nerviosos producidos por esa capa neuronal. Se muestran en la rejilla OUT1 parte inferior mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarVectorOUT2(double OUT2[])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores de las salidas obtenidas de los neurones de la capa de salida y que corresponden a los pulsos nerviosos producidos por esa capa neuronal. Se muestran en la rejilla OUT2 mostrando además una barra de desplazamiento para visualizarlos en su totalidad.

- *MostrarVectorX(double dX[CAPA_ENTRADA])* *Tipo devuelto: void*

  Documentación: Esta función muestra los valores del vector de características del patrón que se está alimentando a la red para entrenarla. Estos valores se muestran en la rejilla X mostrando además una barra de desplazamiento para visualizarlos en su totalidad. .

- *OnButtonEntrena()* *Tipo devuelto: void*

  Documentación: Esta función es la que controla que se cumpla la condición de fin de entrenamiento con la que se dará por terminado el proceso de entrenamiento. Aquí es a donde se lleva el control maestro del proceso de entrenamiento determinando el momento y lugar en donde deberán imprimirse en pantalla los

resultados pertinentes. Esta función es llamada por S.R.P.M. cuando el usuario da clic en el botón de comando “Inicia Entrenamiento RNA” y no se dejará de ejecutarse sino hasta alcanzar la condición de fin de entrenamiento.

- *OnButtonMaster()* *Tipo devuelto: void*

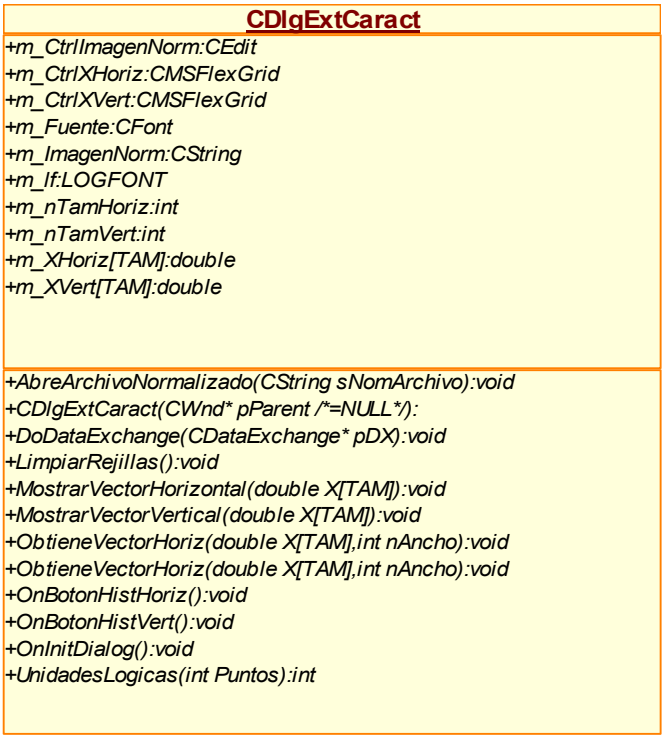
  Documentación: Esta función es llamada por el sistema cuando el usuario da clic en el botón de comando “Selecciona Archivo Maestro” con lo que pide al usuario proporcione el nombre del archivo maestro que hace referencia a los nombres de los archivos de vectores de características y salidas deseadas. Una vez que este archivo ha sido cargado al sistema habilitará el botón de comando “Inicia Entrenamiento RNA”.

- *OnCheckbarrida()* *Tipo devuelto: void*

  Documentación: Esta función miembro es llamada por el esqueleto de trabajo como respuesta a que el usuario haya dado clic en la casilla de verificación “Control barrida a barrida” con lo que habilitará o deshabilitará, según sea el caso, la etiqueta “Ejecutará barrida * barrida” y cuya selección repercutirá directamente en la manera en la que *OnButtonEntrena()* ejecuta su código correspondiente.

- *OnOK()* *Tipo devuelto: void*

  Documentación: Esta función es llamada por el esqueleto de trabajo cuando el usuario da clic al botón de comando “FIN ENTRENAMIENTO” y que hace que se cierre la caja de diálogo “Entrenamiento Red Neuronal”, dando fin al proceso de entrenamiento de la red.



CDlgExtCaract

Descripción: La clase CDlgExtCaract maneja a la caja de diálogo "Histogramas Horizontales y Verticales". Esta caja de diálogo sirve como interfaz de salida para mostrar los resultados obtenido de ejecutar la función *OnOpcionesExtraccindecaractersticas()* de la clase CSRPMView. La caja de diálogo está dividida en tres secciones principales:

La primera corresponde a una caja de texto donde se exhibe la imagen normalizada de la nota musical a la que se le efectuó el proceso de extracción de características. Esta caja de texto está provista de sus barras de desplazamiento para poder visualizar la imagen completa.

La segunda y tercera corresponden a los vectores de características para la red de notas y para la red de tonos, siendo el vector horizontal, el más pequeño, el empleado para los patrones notas y el vertical para los patrones tonos. Ambos vectores fueron obtenidos al aplicar el algoritmo de histogramas verticales y horizontales.

Métodos:

- *AbreArchivoNormalizado(CString sNomArchivo)* *Tipo devuelto: void*

  Documentación: Función miembro que se encarga de abrir el archivo que contiene la información de la imagen normalizada de la nota musical a la que se le aplicó el proceso de extracción de características.

- *LimpiarRejillas()* *Tipo devuelto: void*

  Documentación: Función miembro que borra cualquier dato impreso en las rejillas provistas para exhibir los valores de los vectores de características de la red de notas y de la red de tonos.

- *MostrarVectorHorizontal(double X[TAM])* *Tipo devuelto: void*

  Documentación: Esta función es la que ejecuta los pasos necesarios para visualizar correctamente en la rejilla el vector de características horizontal es decir el de la red de notas.

- *MostrarVectorVertical(double X[TAM])* *Tipo devuelto: void*

  Documentación: Esta función es la que ejecuta los pasos necesarios para visualizar correctamente en la rejilla el vector de características vertical es decir el de la red de tonos.

- *ObtieneVectorHoriz(double X[TAM],int nAncho)* *Tipo devuelto: void*

  Documentación: Esta función obtiene los valores resultado de aplicar la técnica de histogramas horizontales para la obtención de un vector de características de un patrón.

- *ObtieneVectorHoriz(double X[TAM],int nAncho)* *Tipo devuelto: void*

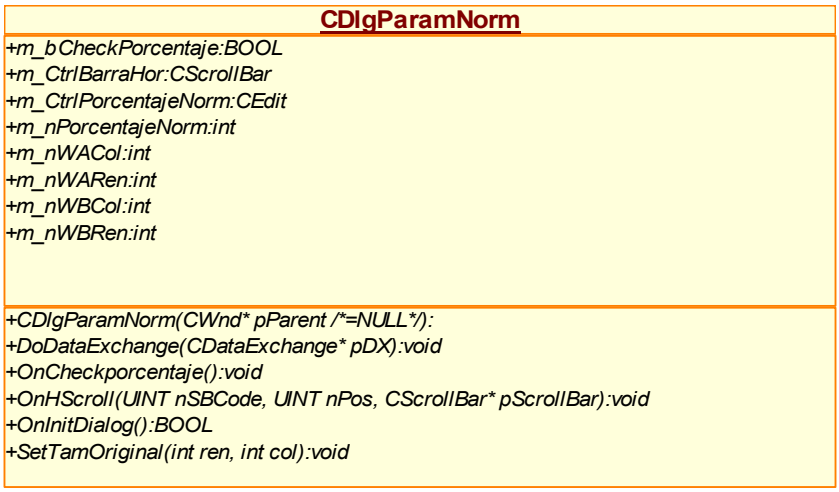
  Documentación: Esta función obtiene los valores resultado de aplicar la técnica de histogramas verticales para la obtención de un vector de características de un patrón.

- *OnBotonHistHoriz()* *Tipo devuelto:* void

Documentación: Función miembro llamada por el esqueleto de trabajo siempre que el usuario de clic en el botón de comando " Calcula Histograma Horizontal" haciendo una llamada a la función miembro *MostrarVectorHorizontal(double X[TAM])*.

- *OnBotonHistVert()* *Tipo devuelto: void*

  Documentación: Función miembro llamada por el esqueleto de trabajo siempre que el usuario de clic en el botón de comando "Calcula Histograma Vertical" haciendo una llamada a la función miembro *MostrarVectorVertical(double X[TAM])*.



CDlgParamNorm

Descripción: Esta clase controla a la caja de diálogo "Parámetros para la normalización", la cual sirve como interfaz de entrada para recibir la definición de los parámetros que deberán utilizarse durante la ejecución de un proceso de normalización de una imagen previamente binarizada. Esta clase entra en ejecución cada vez que el sistema ejecuta la función miembro *OnOpcionesNomalizacin()* de la clase CSRPMView. La caja de diálogo "Parámetros para la normalización" se divide en tres secciones:

La primera denominada "Tamaño imagen original" define el tamaño, en renglones y columnas, de la imagen original binarizada. La segunda denominada "Tamaño imagen

normalizada" define el tamaño, en renglones y columnas, a la que se reducirá la imagen original.

La tercera sección sirve para definir la segunda sección a partir de la determinación de un porcentaje de reducción el cual el usuario puede determinar directamente en la caja de texto o a partir de la barra de desplazamiento que se encuentra provista para este fin.

Estos parámetros son enviados a la función miembro *OnOpcionesNomalizacin()* para entonces efectuar la normalización de la nota musical en turno.

Métodos:

- *OnCheckporcentaje()* *Tipo devuelto: void*

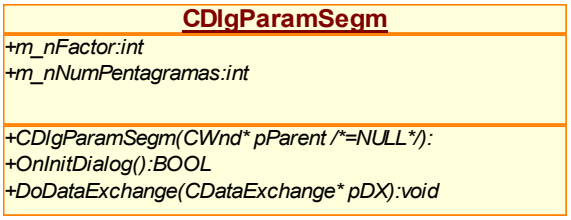
Documentación: Función miembro que es llamada por el esqueleto de trabajo como respuesta a que el usuario haya activado la casilla de verificación "Trabajar con porcentaje" y que activa la caja de edición "Porcentaje de reducción" y la barra de desplazamiento que para este fin se encuentra provista.

- *OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)* *Tipo devuelto: void*

Documentación: Función miembro que controla el movimiento de la barra de desplazamiento y los valores regresados a través de la caja de edición "Porcentaje de reducción". Los movimientos de la barra de desplazamiento pueden ser a través del ratón o por medio del teclado. Se encuentran activadas las teclas: Inicio, Fin, Re Pág y Av Pág.

- *SetTamOriginal(int ren, int col)* *Tipo devuelto: void*

Documentación: Esta función establece los valores de inicio tanto para el tamaño de la imagen original binarizada como los valores sugeridos, de acuerdo a la arquitectura de la red neuronal, para la imagen normalizada.



CDlgParamSegm

Descripción: La clase CDlgParamSegm es utilizada para generar objetos que controlen a la caja de diálogos "Parámetros Segmentación". Dichas cajas de diálogos sirven como interfaz de entrada al usuario para definir solo dos parámetros importantes antes de ejecutar un proceso de segmentación de imágenes digitales. Los parámetros en cuestión son el número de pentagramas que se encuentran en la partitura musical (se considera un solo pentagrama al conjunto de dos pentagramas unidas uno bajo clave de sol y el otro bajo clave de fa) y el factor de corrección el cual juega el mismo papel que el ya descrito en el capítulo 3 para la binarización de notas musicales.

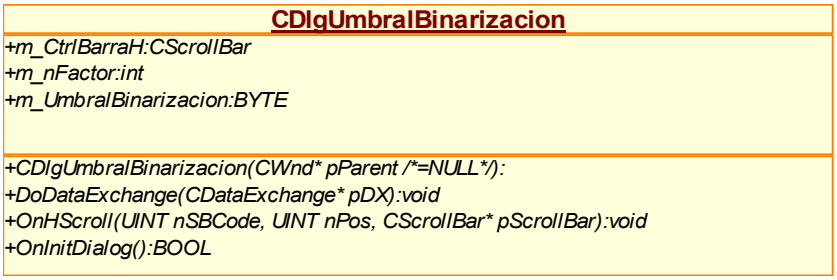
Atributos:

- *m_nFactor* *Tipo devuelto: int*

Documentación: Variable miembro que guarda el valor del factor de correción sugerido por el usuario para el correcto procesamiento de la partitura musical.

- *m_nNumPentagramas* *Tipo devuelto: int*

Documentación: Variable miembro que guarda el número de pentagramas que se encuentran en la partitura que está por segmentarse. Cabe mencionar que el número de pentagramas se debe contabilizar tomando en cuenta que aquellas partituras que muestran pentagramas con caracteres musicales escritos en dos claves (sol y fa), deberán contabilizarse como un solo pentagrama.



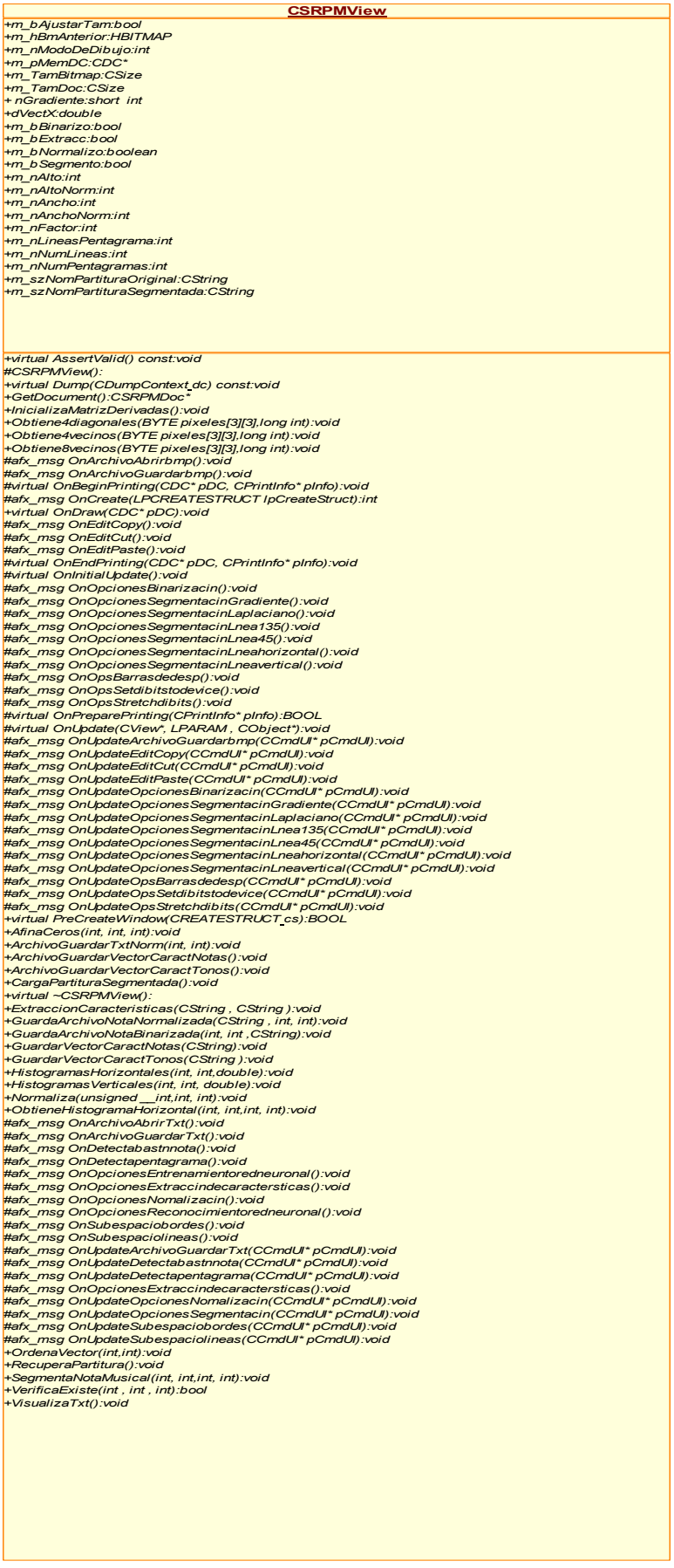
CDlgUmbralBinarizacion

Descripción: Esta clase da servicio a la caja de diálogo "Valor umbral para RGB píxeles", que sirve como interfaz de entrada de parámetros iniciales para un procesamiento de imágenes de mapas de bits como puede ser un proceso de segmentación o un proceso de binarización de imágenes. La caja de diálogos consta básicamente de dos parámetros: el factor de corrección cuyo significado es muy similar al ya explicado en la clase CDlgParamSegm y el valor umbral RGB (0-255) y que servirá como criterio a los procedimientos antes mencionados para realizar algún tipo de clasificación a cada uno de los píxeles que conforman a una imagen en mapa de bits.

Métodos:

- OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar). Tipo devuelto: void

Documentación: Función miembro que controla el movimiento de la barra de desplazamiento y los valores regresados a través de la caja de edición "Porcentaje de reducción". Los movimientos de la barra de desplazamiento pueden ser a través del ratón o por medio del teclado. Se encuentran activadas las teclas: Inicio, Fin, Re Pág y Av Pág.


CSRPMView
+m_bAjustarTam:bool
+m_hBmAnterior:HBITMAP
+m_nModoDeDibujo:int
+m_pMemDC:CDC*
+m_TamBitmap:CSize
+m_TamDoc:CSize
+ nGradiente:short int
+dVectX:double
+m_bBinarizo:bool
+m_bExtracc:bool
+m_bNormalizo:boolean
+m_bSegmento:bool
+m_nAlto:int
+m_nAltoNorm:int
+m_nAncho:int
+m_nAnchoNorm:int
+m_nFactor:int
+m_nLineasPentagrama:int
+m_nNumLineas:int
+m_nNumPentagramas:int
+m_szNomPartituraOriginal:CString
+m_szNomPartituraSegmentada:CString
+virtual AssertValid() const:void
#CSRPMView():
+virtual Dump(CDumpContext_dc) const:void
+GetDocument():CSRPMDoc*
+InicializaMatrizDerivadas():void
+Obtiene4diagonales(BYTE pixeles[3][3],long int):void
+Obtiene4vecinos(BYTE pixeles[3][3],long int):void
+Obtiene8vecinos(BYTE pixeles[3][3],long int):void
#afx_msg OnArchivoAbrirbmp():void
#afx_msg OnArchivoGuardarbmp():void
#virtual OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo):void
#afx_msg OnCreate(LPCREATESTRUCT lpCreateStruct):int
+virtual OnDraw(CDC* pDC):void
#afx_msg OnEditCopy():void
#afx_msg OnEditCut():void
#afx_msg OnEditPaste():void
#virtual OnEndPrinting(CDC* pDC, CPrintInfo* pInfo):void
#virtual OnInitialUpdate():void
#afx_msg OnOpcionesBinarizacin():void
#afx_msg OnOpcionesSegmentacinGradiente():void
#afx_msg OnOpcionesSegmentacinLaplaciano():void
#afx_msg OnOpcionesSegmentacinLnea135():void
#afx_msg OnOpcionesSegmentacinLnea45():void
#afx_msg OnOpcionesSegmentacinLneahorizontal():void
#afx_msg OnOpcionesSegmentacinLneavertical():void
#afx_msg OnOpsBarrasdedesp():void
#afx_msg OnOpsSetdibitstodevice():void
#afx_msg OnOpsStretchdibits():void
#virtual OnPreparePrinting(CPrintInfo* pInfo):BOOL
#virtual OnUpdate(CView*, LPARAM , CObject*):void
#afx_msg OnUpdateArchivoGuardarbmp(CCmdUI* pCmdUI):void
#afx_msg OnUpdateEditCopy(CCmdUI* pCmdUI):void
#afx_msg OnUpdateEditCut(CCmdUI* pCmdUI):void
#afx_msg OnUpdateEditPaste(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesBinarizacin(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacinGradiente(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacinLaplaciano(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacinLnea135(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacinLnea45(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacinLneahorizontal(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacinLneavertical(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpsBarrasdedesp(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpsSetdibitstodevice(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpsStretchdibits(CCmdUI* pCmdUI):void
+virtual PreCreateWindow(CREATESTRUCT_cs):BOOL
+AfinaCeros(int, int, int):void
+ArchivoGuardarTxtNorm(int, int):void
+ArchivoGuardarVectorCaractNotas():void
+ArchivoGuardarVectorCaractTonos():void
+CargaPartituraSegmentada():void
+virtual ~CSRPMView():
+ExtraccionCaracteristicas(CString , CString):void
+GuardaArchivoNotaNormalizada(CString , int, int):void
+GuardaArchivoNotaBinarizada(int, int ,CString):void
+GuardarVectorCaractNotas(CString):void
+GuardarVectorCaractTonos(CString):void
+HistogramasHorizontales(int, int,double):void
+HistogramasVerticales(int, int, double):void
+Normaliza(unsigned __int,int, int):void
+ObtieneHistogramaHorizontal(int, int,int, int):void
#afx_msg OnArchivoAbrirTxt():void
#afx_msg OnArchivoGuardarTxt():void
#afx_msg OnDetectabastnnota():void
#afx_msg OnDetectapentagrama():void
#afx_msg OnOpcionesEntrenamientoredneuronal():void
#afx_msg OnOpcionesExtraccindecaracteristicas():void
#afx_msg OnOpcionesNomalizacin():void
#afx_msg OnOpcionesReconocimientoredneuronal():void
#afx_msg OnSubespaciobordes():void
#afx_msg OnSubespaciolineas():void
#afx_msg OnUpdateArchivoGuardarTxt(CCmdUI* pCmdUI):void
#afx_msg OnUpdateDetectabastnnota(CCmdUI* pCmdUI):void
#afx_msg OnUpdateDetectapentagrama(CCmdUI* pCmdUI):void
#afx_msg OnOpcionesExtraccindecaracteristicas():void
#afx_msg OnUpdateOpcionesNomalizacin(CCmdUI* pCmdUI):void
#afx_msg OnUpdateOpcionesSegmentacin(CCmdUI* pCmdUI):void
#afx_msg OnUpdateSubespaciobordes(CCmdUI* pCmdUI):void
#afx_msg OnUpdateSubespaciolineas(CCmdUI* pCmdUI):void
+OrdenaVector(int,int):void
+RecuperaPartitura():void
+SegmentaNotaMusical(int, int,int, int):void
+VerificaExiste(int , int , int):bool
+VisualizaTxt():void

CSRPMView

Descripción: Las responsabilidades de esta clase son desplegar los datos del documento gráficamente al usuario y aceptar e interpretar las entradas del usuario como operaciones en el documento. Las tareas principales al escribir esta clase derivada de CScrollView son:

- Escribir la función miembro OnDraw que redibuja los datos del documento en la ventana de vista
- Conectar los mensajes de Windows apropiados a los objetos que proporcionan servicios de interface con el usuario tales como elementos de menú, a funciones manejadoras de mensajes en la clase de vista.
- Implementar esos manejadores para interpretar las entradas de los usuarios.

Esta clase contiene las funciones específicas del S.R.P.M. siendo las más importantes la segmentación a través del criterio del gradiente y del Laplaciano, la binarización, la normalización de Güdsen, la extracción de características y los reconocedores neuronales.

Métodos:

- *GetDocument()* *Tipo devuelto: CSRPMDoc **

Documentación: Esta función regresa un apuntador al documento al que está asociada la vista de la aplicación.

- *InicializaMatrizDerivadas()* *Tipo devuelto: void*

Documentación: Esta función miembro pone en cero todas las matrices a ser empleadas por SRPM.

- *Obtiene4diagonales(BYTE pixeles[3][3],long int num_pixel)* *Tipo devuelto: void*

Documentación: Esta función obtiene los píxeles $N_D(p)$ del píxel que se está analizando mediante alguna de las máscaras de Sobel, para mayor información sobre los píxeles $N_D(p)$ ver 2.3.

- *Obtiene4vecinos(BYTE pixeles[3][3],long int num_pixel)* *Tipo devuelto: void*

Documentación: Esta función obtiene los píxeles $N_4(p)$ del píxel que se está analizando mediante alguna de las máscaras de Sobel, para mayor información sobre los píxeles $N_4(p)$ ver 2.3.

- *Obtiene8vecinos(BYTE pixeles[3][3],long int num_pixel)* *Tipo devuelto: void*

Documentación: Esta función obtiene los píxeles ocho vecinos del píxel determinado por el parámetro num_pixel. Los píxeles 8 vecinos son obtenidos por llamadas sucesivas a las funciones miembro Obtiene4diagonales y Obtiene4vecinos. Los $N_8(p)$ del píxel que se está analizando están determinados por los $N_4(p) + N_D(p)$, para mayor información sobre los píxeles $N_D(p)$ ver 2.3.

- *afx_msg OnArchivoAbrirbmp();* *Tipo devuelto: void*

Documentación: El propósito de esta función miembro es crear un mapa de bits desde un archivo *.bmp; es decir, leer un archivo que contenga un mapa de bits independiente de dispositivo, normalmente con extensión *.bmp y cargarlo en el objeto CDIB del documento. A continuación a partir de este objeto se crea un mapa de bits dependiente de dispositivo y se selecciona en un contexto de dispositivo memoria, apuntado por el dato miembro m_pMemDC, y definido en la vista. Es más eficiente manipular el mapa de bits (partitura musical digitalizada) desde un DDB que desde un DIB, por que la conversión de colores ya está hecha.

- *afx_msg OnArchivoGuardarbmpcomo();* *Tipo devuelto: void*

Documentación: Cuando se ejecuta la orden “Guardar partitura como…” del menú Archivo, el esqueleto de trabajo de S.R.P.M. llama a ésta función miembro solicitándole al usuario el nombre del archivo e invocando también a la función Serialize del documento, para permitir guardarlos datos en el archivo especificado.

Por otra parte, cuando el documento haya sido modificado, es bueno avisar al usuario si éste intenta salir de la aplicación sin guardar dicho documento. Para ello, cada vez que el documento sea modificado se invocará a la función SetModifiedFlags del documento con el fin de poner en verdadero el indicador de modificado. También, es

bueno mantener la orden “Guardar partitura como…” inactiva mientras el documento no haya sido modificado y activa en caso contrario.

- *OnDraw(CDC* pDC)* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo de S.R.P.M. para redibujar las partituras y/o datos del documento. El esqueleto de trabajo llama a esta función miembro cuando se tiene que redibujar la pantalla, imprimir o realizar un proceso de previsualización de la impresión recibiendo difrenetes contextos de dispositivo en cada caso. No existe una implementación por default de esta función aún cuando se encuentra heredada directamente de la clase CView. Entre muchas de las actividades que esta función desempaña están el pintar el mapa de bits que contiene la partitura a ser procesada a partir de la esquina superior izquierda de la vista. Esto se hace utlizando:

  - StretchDIBits.- Este es el modo de dibujo elegido por omisión y es fijado por la variable miembro m_nModoDeDibujo que es iniciada a 1 en el constructor de la vista.
  - SetDIBitsToDevice.- Utilizando esta opción se obtiene el mismo resultado tanto si hay barras de desplazamiento como si se hubiese establecido el modo de “ampliar o reducir hasta ajustar”. El mapa de bits se obtiene directamente del objeto CDIB.

- *afx_msg OnOpcionesBinarizacin()* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado del menú “Opciones” la opción “Binarización”. La función crea un objeto perteneciente a la clase CDlgUmbralBinarización y es mediante este objeto que el sistema le solicita al usuario determine el umbral RGB para píxeles así como el factor de corrección a aplicar al DIB. Se accede mediante un apuntador al mapa de bits del DIB y entonces se clasifica píxel a píxel como uno o como cero dependiendo de si el valor RGB del píxel en turno es mayor o menor que el valor umbral determinado por el objeto de la clase CDlgUmbralBinarización. Estos números se van almacenando en la variable miembro nGradiente

Una vez que el proceso se ha realizado para todos ls píxeles del DIB, la función hace una llamada a la función miembro *OnArchivoGuardarTxt()* y se vacía la información contenida en nGradiente a un archivo de texto cuyo nombre lo determina el usuario. Acto seguido la función crea un objeto perteneciente a la clase CDlgBinariza para poder mostrar a través de él, en pantalla, la imagen recién binarizada.

- *afx_msg OnOpcionesSegmentacinGradiente()* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo cuando el usuario ha seleccionado del menú "Utilerías" la opción "Máscaras de detección" y del submenú emergente la opción "Gradiente". Esta función ejecuta un mascareo del DIB empleando la máscara para cálculo del gradiente de una imagen. Ver fig 2.7.

- *afx_msg OnOpcionesSegmentacinLaplaciano()* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo cuando el usuario ha seleccionado del menú "Utilerías" la opción "Máscaras de detección" y del submenú emergente la opción "Laplaciano". Esta función ejecuta un mascareo del DIB empleando la máscara para cálculo del Laplaciano de una imagen y que afina los resultados obtenidos por un mascareo previo del gradiente de la imagen. Ver fig 2.9.

- *afx_msg OnOpcionesSegmentacinLnea135()* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo cuando el usuario ha seleccionado del menú "Utilerías" la opción "Máscaras de detección" y del submenú emergente la opción "Línea -45°". Esta función ejecuta un mascareo del DIB empleando la máscara apropiada para la detección de líneas en la imagen con la característica indicada.

- *afx_msg OnOpcionesSegmentacinLnea45()* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo cuando el usuario ha seleccionado del menú "Utilerías" la opción "Máscaras de detección" y del submenú emergente la opción "Línea 45°". Esta función ejecuta un mascareo del DIB empleando la máscara apropiada para la detección de líneas en la imagen con la característica indicada.

- *afx_msg OnOpcionesSegmentacinLneahorizontal()* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por el esqueleto de trabajo cuando el usuario ha seleccionado del menú "Utilerías" la opción "Máscaras de detección" y del submenú emergente la opción "Línea Horizontal". Esta función ejecuta un mascareo del DIB empleando la máscara apropiada para la detección de líneas en la imagen con la característica indicada. Esta función es la llamada por el procedimiento de detección de líneas de pentagrama para su oportuna detección durante la segmentación de una partitura musical.

- *afx_msg OnOpcionesSegmentacinLneavertical()* *Tipo devuelto:* void

Documentación: Esta función miembro es llamada por el esqueleto de trabajo cuando el usuario ha seleccionado del menú "Utilerías" la opción "Máscaras de detección" y del submenú emergente la opción "Línea Vertical". Esta función ejecuta un mascareo del DIB empleando la máscara apropiada para la detección de líneas en la imagen con la característica indicada. Esta función es la llamada por el procedimiento de segmentación de acordes y corcheas en la clase CDlgReconocimiento.

- *afx_msg OnOpsBarrasdedesp()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado la opción "Barras de desplazamiento" del menú "Opciones". El efecto de esta función es la aparición de barras de desplazamiento horizontales y verticales en la ventana de vista principal para poder visualizar completamente la imagen de una partitura cuando esta excede en dimensiones a las dimensiones del area de trabajo de la ventana de vista.

- *afx_msg OnOpsSetdibitstodevice()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado la opción "Visualiza DIB (estirando/comprimiendo)" del menú "Opciones". El efecto de la ejecución de esta función miembro es que se muestra, en el área de trabajo de la ventana de vista, el DIB previamente cargado mediante la opción "Abrir imagen" del menú "Archivo" pero forzando mediante estiramiento o compresión de píxeles a que la imagen ocupe dicha área de trabajo.

- *afx_msg OnOpsStretchdibits()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado la opción "Visualiza DIB (tamaño area de trabajo)" del menú "Opciones". El efecto de la ejecución de esta función miembro es que se muestra, en el área de trabajo de la ventana de vista, el DIB previamente cargado mediante la opción "Abrir imagen" del menú "Archivo". El DIB ocupará un área exactamente igual a sus dimensiones.

- *AfinaCeros(int, int, int)* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *SegmentaNotaMusical()*, y sirve para afinar la ubicación de objetos que están ligeramente separados de otros y que en realidad conforman un solo objeto. Como ejemplo de lo anterior, podemos citar a la clave de fa, en donde los dos puntos están ligeramente separados del resto de la figura pero que en conjunto conforman un solo objeto: la clave de fa. La función recibe como parámetros el vector donde se ubican los ceros y que identifican los objetos que previamente fueron detectados por el procedimiento de segmentación de partituras musicales, un valor entero que indica la tolerancia de separación entre objetos sin que esto implique objetos distintos y el tamaño que tiene el vector donde se ubicaron los ceros identificadores de objetos.

- *ArchivoGuardarTxtNorm(int, int)* *Tipo devuelto: void*

Documentación: Función miembro que es llamada por la función *OnOpcionesNomalizacin(),* para guardar en un archivo de texto la imagen de la nota musical recién normalizada.

- *ArchivoGuardarVectorCaractNotas()* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *OnOpcionesExtraccindecaractersticas()*, para guardar en un archivo de texto el vector de características obtenido por el método de histogramas verticales y que constituye la entrada a la red de notas.

- *ArchivoGuardarVectorCaractTonos()* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *OnOpcionesExtraccindecaractersticas()*, para guardar en un archivo de texto el vector de características obtenido por el método de histogramas horizontales y que constituye la entrada a la red de tonos.

- *CargaPartituraSegmentada()* *Tipo devuelto: void*

Documentación: Función miembro empleada para cargar en el área de trabajo a la partitura segmentada.

- *ExtraccionCaracteristicas(CString , CString)* *Tipo devuelto: void*

Documentación: Esta función miembro extrae las caraterísticas de una nota musical normalizada aplicando el método de los histogramas verticales y horizontales y obteniendo como resultado dos vectores: uno que servirá como vector de características para la red de notas y otro que fungirá como vector de características para la red de tonos. Ambos vectores de características son guardados en archivos de texto.

- *GuardaArchivoNotaNormalizada(CString , int, int)* *Tipo devuelto: void*

Documentación: Esta función es llamada por la función miembro *SegmentaNotaMusical(),* una vez que se ha segmentado, binarizado y normalizado una nota musical del pentagrama. El sistema necesitará guardar los resultados del proceso de normalización en un archivo de texto cuyo nombre será dado automáticamente por el propio sistema.

- *GuardaArchivoNotaBinarizada(int, int ,CString)* *Tipo devuelto:* void

Documentación: Esta función es llamada por la función miembro *SegmentaNotaMusical(),* una vez que se ha segmentado y binarizado una nota musical del pentagrama. El sistema necesitará guardar los resultados del proceso de binarización en un archivo de texto cuyo nombre será dado automáticamente por el propio sistema.

- *GuardarVectorCaractNotas(CString)* *Tipo devuelto: void*

Documentación: Esta función es llamada por la función miembro *SegmentaNotaMusical(),* una vez que se ha segmentado, binarizado, normalizado y extraído las características de una nota musical del pentagrama. El sistema necesitará guardar los resultados del proceso de extracción de características mediante histogramas verticales en un archivo de texto cuyo nombre será dado automáticamente por el propio sistema.

- *GuardarVectorCaractTonos(CString)* *Tipo devuelto: void*

Documentación: Esta función es llamada por la función miembro *SegmentaNotaMusical(),* una vez que se ha segmentado, binarizado, normalizado y extraído las características de una nota musical del pentagrama. El sistema necesitará guardar los resultados del proceso de extracción de características mediante histogramas horizontales en un archivo de texto cuyo nombre será dado automáticamente por el propio sistema.

- *HistogramasHorizontales(int, int,double)* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *ExtraccionCaracteristicas()*, que calcula el vector de características de una nota musical mediante el algoritmo de histogramas horizontales en el cual se suman los bits por renglón de la nota normalizada y al final se divide ese resultado entre el ancho de la imagen normalizada obteniendo un número representativo por cada renglón que tenga dicha imagen y que servirá como su vector de características.

- *HistogramasVerticales(int, int, double)* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *ExtraccionCaracteristicas()*, que calcula el vector de características de una nota musical mediante el algoritmo de histogramas verticales en el cual se suman los bits por columna de la nota normalizada y al final se divide ese resultado entre el alto de la imagen normalizada obteniendo un número representativo por cada columna que tenga dicha imagen y que servirá como su vector de características.

- *Normaliza(unsigned __int,int, int)* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por la función *OnOpcionesNomalizacin()* consecutivamente logrando en cada llamada la normalización de un renglón o una columna de la imagen binarizada. Esta función es la codificación del algoritmo de Normalización de Güdsen.

- *ObtieneHistogramaHorizontal(int, int,int, int)* *Tipo devuelto: void*

Documentación: Esta función miembro es llamada por la función *OnOpcionesSegmentacin()* de la misma clase CSRPMView y cuyo propósito es el de obtener una contabilización por columnas de los bits que componen a un pentagrama dado que se encuentra dentro de una partitura musical a la cual se le detectaron y eliminaron ya todas las líneas de los pentagramas que la componen. El vector resultante servirá como fuente de información base para la determinación del número y ubicación.

- *afx_msg OnArchivoAbrirTxt()* *Tipo devuelto: void*

Documentación: Esta función miembro abre un archivo de texto que representa la imagen de una nota binarizada o normalizada.

- *afx_msg OnArchivoGuardarTxt()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado la opción “Guardar como… Archivo texto” del menú “Archivo”. Esta función guarda en un archivo de texto los datos correspondientes a una imagen binarizada o normalizada.

- *afx_msg OnDetectabastnnota()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado la opción “Detecta Bastón Nota” del menú “Utilerias” y cuya misión es la de detectar los bastones de corcheas o de acordes para su posterior segmentación.

- *afx_msg OnDetectapentagrama()* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *OnOpcionesSegmentacin()* y cuyo objetivo es el detectar la ubicación de todas las líneas de todos los pentagramas que componen a una partitura musical. La detección se logra gracias al uso de la máscara de detección de líneas horizontales y al análisis de histogramas verticales que se practican sobre el DIB que representa a la partitura musical que se intenta segmentar.

- *afx_msg OnOpcionesEntrenamientoredneuronal()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado la opción "Entrenamiento red neuronal" del menú "Opciones". La función tan solo crea un objeto de la clase CDlgEntrenaRNA y deja que dicha clase se encargue de lo demás.

- *afx_msg OnOpcionesExtraccindecaractersticas()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haga una selección de "Extracción de características" del menú "Opciones". La función trabaja con la imagen recién normalizada obteniendo sus histogramas horizontales y verticales que a su vez se convertirán en los vectores de características de la nota musical, uno para la red de notas y el otro para la red de tonos, dichos vectores son guardados oportunamente en archivos de texto para su futuro uso.

- *afx_msg OnOpcionesNomalizacin()* *Tipo devuelto: void*

Documentación: Función miembro que es llamada por el esqueleto de trabajo como respuesta a una acción del usuario en la que haya seleccionado la opción "Normalización" del menú "Opciones. La función manipula a la imagen binarizada mediante el algoritmo de normalización de Güdsen tomando en cuenta los parámetros aportados por el usuario a través de la caja de diálogo "Parámetros para la normalización" y que ya se explicaron en la documentación de la clase CDlgParamNorm". Los resultados de la normalización de Güdsen son guardados oportunamente en un archivo de texto cuyo nombre debe ser proporcionado por el usuario.

- *afx_msg OnOpcionesReconocimientoredneuronal()* *Tipo devuelto: void*

Documentación: Función miembro que es llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado “Reconocimiento red neuronal” en el menú “Opciones”. La función tan solo crea un objeto de la clase CDlgReconocimiento y deja que dicha clase se encargue de lo demás

- *afx_msg OnSubespaciobordes()* *Tipo devuelto: void*

Documentación: Función miembro que es llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado “Detección combinada… SubEspacioBordes” en el menú “Utilerías”. La función determina cuales de los píxeles que se encuentran en un DIB pertenecen más al subespacio de los bordes que al subespacio de las líneas. Este tipo de detección afina detecciones previas hechas por medio de otras máscaras de detección.

- *afx_msg OnSubespaciolineas()* *Tipo devuelto: void*

Documentación: Función miembro que es llamada por el esqueleto de trabajo como respuesta a que el usuario haya seleccionado “Detección combinada… SubEspacioLíneas” en el menú “Utilerías”. La función determina cuales de los píxeles que se encuentran en un DIB pertenecen más al subespacio de las líneas que al subespacio de los bordes. Este tipo de detección afina detecciones previas hechas por medio de otras máscaras de detección.

- *OrdenaVector(int,int)* *Tipo devuelto: void*

Documentación: Función miembro llamada por la función *OnOpcionesSegmentacin(),* que sirve para ordenar los valores contenidos en el vector que guarda la contabilización de los valores RGB por renglón de toda la partitura musical y con lo que se intentará más tarde ubicar la posición de las líneas de todos los pentagramas que componen a la partitura.

- *RecuperaPartitura()* *Tipo devuelto: void*

Documentación: Función miembro que sirve para volver a restaurar a la partitura musical, a la cual se le está aplicando el proceso de segmentación, a su estado original.

Esta función miembro es llamada por la función *SegmentaNotaMusical()* de la misma clase CSRPMView.

- *SegmentaNotaMusical(int, int,int, int)* *Tipo devuelto: void*

Documentación: Esta función es llamada por la función miembro *OnOpcionesSegmentacin(),* y prácticamente es la responsable de llevar a cabo el proceso de segmentación de un pentagrama de la partitura musical.

La función comienza calculando el número de objetos que se encuentran en la partitura con la ayuda del vector nUbicaCeros y guardando esa información en la variable nNumObjetos. Esta información es refinada por la ejecución de la función *AfinaCeros()* la cual ya se ha explicado con anterioridad, una vez hecho esto la función *Segmenta NotaMusical()* calcula las posiciones en las que se encuentran cada objeto del pentagrama en procesamiento. Se asignan nombres automáticamente para los archivos que contendrán a las imágenes binarizada, normalizada, vector de características para red de notas y para red de tonos respectivamente mediante llamadas sucesivas a las funciones miembro *OnOpcionesBinarizacin(), OnOpcionesNomalizacin(), ExtraccionCaracteristicas()*.

- *VerificaExiste(int , int , int)* *Tipo devuelto: bool*

Documentación: Esta función miembro es llamada por la función *OnDetectapentagrama(),* cuando ésta necesita verificar si una cierta línea de pentagrama ya ha sido detectada con anterioridad o no.

CDlgReconocimiento
+m_bSostenido:bool +m_cClave:char +m_CRedNotas:CRNA +Tonos m_CRedTonos:CRNA +m_CtrlArchNotas:CButton +m_CtrlArchTonos:CButton +m_CtrlFigura:CEdit +m_CtrlPartitura:CButton +m_CtrlReconoceNota:CButton +m_CtrlReconoceTono:CButton +m_CtrlSiguientePatron:CButton +m_CtrlVectorNotas:CMSFlexGrid +m_CtrlVectorTonos:CMSFlexGrid +m_Fuente:CFont +m_lf:LOGFONT +m_nNumPatron:int +m_nNumTonos:int +m_sArchNotas:CString +m_sArchPartitura:CString +m_sArchPatronNota:CString +m_sArchPatronTono:CString +m_sArchTonos:CString +m_sNomArchNotas:CString +m_sNomArchSalidasDeseadasNotas:CString +m_sNomArchSalidasDeseadasTonos:CString +m_sNomArchTonos:CString +m_sNotaNormalizada:CString +m_sNotaReconocida:CString +m_sTonoReconocido:CString
+AbreArchivoNormalizado(CString sNomArchivo):void +CDlgReconocimiento(CWnd* pParent /*=NULL*/): +DeterminaCualNota(int nCual):void +DeterminaCualTonoClaveFa(int nCual):void +DeterminaCualTonoClaveSol(int nCual):void +DistanciaLineasPentagrama(int nVector[]):int +DoDataExchange(CDataExchange* pDX):void +GeneraVectCaractTonoAcorde(int nNumBola, CString sNotaNorm):CString +GuardaArchivoTxt():void +GuardaVector(double X[], int nNumBola):CString +MostrarVectorNotas(double X[]):void +MostrarVectorTonos(double X[]):void +ObtieneVectorCaract(double X[], int nBaston, int nLineasPentagrama[], int inicio, int fin, CString sAcorde):void +OnButtonnotas():void +OnButtonpartitura() :void +OnButtonreconoceNota():void +OnButtonreconoceTono():void +OnButtonsigPatron():void +OnButtontonos() :void +OnInitDialog() :BOOL +SetNombreArchivoFigura(CString sNombre):CString +UnidadesLogicas(int Puntos):int +VerificaExiste(int nVector[], int nNum):bool

CDlgReconocimiento

Descripción: Esta clase es la responsable de efectuar el proceso de reconocimiento de cada una de las notas musicales que componen a una partitura. Este proceso de reconocimiento proporciona: el tiempo que debe tocarse cada nota musical debido a su forma (negrita, redonda, blanca, etc.) así como que nota es de acuerdo con la posición

que guarda en el pentagrama. También determina en clave se está tocando la melodía (clave de sol o clave de fa).

Esta clase es la responsable de manipular la caja de diálogo “Reconocimiento Notas Musicales” la cual está dividida en tres partes:

La primera se denomina: “Datos Redes Neuronales Entrenadas”, en esta parte el usuario define pulsando en los botones de comando provistos para este fin los archivos de texto que contienen los pesos y umbrales finales de las redes neuronales de tonos y notas entrenadas previamente.

La segunda se denomina: “Datos Partitura a Reconocer:”, en esta parte el usuario define el nombre del archivo que contiene los datos de la partitura generados por el proceso de segmentación. Este archivo es cargado al sistema a través del botón de comando “Selecciona Archivo Partitura” el cual quedá acvtivado después de haber cargado los archivos de pesos y umbarles de las dos redes neuronales con las que se trabajará.

La tercera parte se denomina: “Reconocimiento del patrón” y es la que sirve como interface de entrada y salida con el usuario guíando el proceso de reconocimiento de notas musicales que componen a una cierta partitura. Existe una caja de texto donde se despliega la imagen normalizada de la nota musical en turno a ser reconocida junto con dos rejillas donde se despliegan cada uno de los vectores de características que se obtuvieron con anterioridad de la mencionada nota. En el lado derecho de esta parte existen tres botones de comando junto con tres cajas de edición. Los dos primeros botones de comando sirven para hacer que las redes (de notas y tonos respectivamente) reconozcan los vectores de características dados, las cajas de edición dan la conclusión alcanzada por cada red con respecto a la nota musical (en cuanto a su forma y posición en pentagrama). El botón de comando de hasta abajo etiquetado como “Siguiente Patrón” es pulsado por el usuario una vez que ambas redes han alcanzado alguna conclusión sobre la nota en turno.

Métodos:

- *AbreArchivoNormalizado(CString sNomArchivo)* *Tipo devuelto: void*

Documentación: Esta función es llamada cada vez que se necesita cargar la imagen normalizada de la nota musical que se intenta reconocer.

- *CDlgReconocimiento(CWnd* pParent /*=NULL*/)* *Tipo devuelto: No hay tipo disponible.*

Documentación: Constructor por default de la clase CDlgReconocimiento.

- *DeterminaCualNota(int nCual)* *Tipo devuelto: void*

Documentación: Función miembro que determina que carácter musical es el que reconoció la red de notas. Esta función es llamada por la función miembro *OnButtonreconoceNota()*.

- *DeterminaCualTonoClaveFa(int nCual)* *Tipo devuelto: void*

Documentación: Función miembro que determina que tono musical en clave de Fa es el que reconoció la red de tonos. Esta función es llamada por la función miembro *OnButtonreconoceTono()* y determina que código MIDI es el que le corresponde al tono reconocido.

- DeterminaCualTonoClaveSol(int nCual) Tipo devuelto: void

Documentación: Función miembro que determina que tono musical en clave de Sol es el que reconoció la red de tonos. Esta función es llamada por la función miembro *OnButtonreconoceTono()* y determina que código MIDI es el que le corresponde al tono reconocido.

- *DistanciaLineasPentagrama(int nVector[])* *Tipo devuelto: int*

Documentación: Función miembro que determina a que distancia de encuentran equidistantemente cada una de las líneas del pentagrama en el que se encuentra dibujado el carácter musical en turno.

- *GeneraVectCaractTonoAcorde(int nNumBola, CString sNotaNorm) Tipo devuelto: CString*

Documentación: Función miembro llamada por la función OnButtonreconoceTono(), y que es la responsable de segmentar a un acorde dependiendo de la conclusión que envía la red de notas.

- *GuardaArchivoTxt() Tipo devuelto: void*

Documentación: Función miembro que se llama para guardar la imagen normalizada y segmentada de un acorde o de una corchea de más de un bastón.

- *GuardaVector(double X[], int nNumBola) Tipo devuelto: CString*

Documentación: Función miembro que se llama pára guardar el vector de características del acorde o corchea segmentada para después emplearlo en la red de tonos.

- *MostrarVectorNotas(double X[]) Tipo devuelto: void*

Documentación: Función miembro que se emplea para llenar la rejilla horizontal con los datos del vector de características del carácter musical a ser alimentado a la red de notas.

- *MostrarVectorTonos(double X[]) Tipo devuelto: void*

Documentación: Función miembro que se emplea para llenar la rejilla vertical con los datos del vector de características del carácter musical a ser alimentado a la red de tonos.

- *ObtieneVectorCaract(double X[], int nBaston, int nLineasPentagrama[], int inicio, int fin, CString sAcorde) Tipo devuelto: void*

Documentación: Función miembro que obtiene el vector de características por el método de histogramas verticales y horizontales de un acorde o de una corchea recién segmentada.

- *OnButtonnotas()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya dado clic al botón de comando "Selecciona Archivo de Pesos RNA Notas Musicales" y que proporciona al objeto m_CRedNotas los pesos y umbrales con los que deberá trabajar.

- *OnButtonpartitura()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya pulsado el botón de comando "Selecciona Archivo Partitura" y que carga al sistema los datos necesarios para que S.R.P.M. vaya leyendo uno a uno los caracteres musicales que componen a la partitura para que puedan ser completamente reconocidos.

- *OnButtonreconoceNota()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo cuando el usuario ha pulsado el botón de comando "RECONOCE Nota Musical" y que proporciona al objeto m_CRedNotas el vector de características del carácter musical en turno para que pueda ser clasificado en base a que nota es.

- *OnButtonreconoceTono()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo cuando el usuario ha pulsado el botón de comando "RECONOCE Tono Musical" y que proporciona al objeto m_CRedTonos el vector de características del carácter musical en turno para que pueda ser clasificado en base a que tono es.

- *OnButtonsigPatron()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo cuendo el usuario ha pulsado el botón de comando "Siguiente Patrón" y que provoca que el sistema lea los datos del siguiente carácter musical a ser reconocido por las redes.

- *OnButtontonos()* *Tipo devuelto: void*

Documentación: Función miembro llamada por el esqueleto de trabajo como respuesta a que el usuario haya dado clic al botón de comando “Selecciona Archivo de Pesos RNA Tonos Musicales” y que proporciona al objeto m_CRedTonos los pesos y umbrales con los que deberá trabajar.

- *SetNombreArchivoFigura(CString sNombre)* *Tipo devuelto: CString*

Documentación: Establece el nombre del archivo de texto que tiene la imagen normalizada del carácter musical ha reconocer.