Capítulo 4

Implementación del editor de percusiones.

4.1 Consideraciones para la implementación.

Para la implementación de la aplicación, se hicieron varias consideraciones. La primera de ellas, es el aspecto del tiempo, ya que la aplicación depende en gran medida del manejo de éste. Por este motivo, se tomó la decisión de implementar el sistema en Turbo Delphi, ya que de entre todas las herramientas con las que cuenta, se encontró una muy útil, llamada TTimer, la cual es en la que se basa este proyecto para llevar a cabo la reproducción de los patrones.

Otra consideración que es importante mencionar, es la relacionada con el manejo de las señales MIDI. Para poder mandar las señales MIDI de salida, se utilizó un método de un módulo utilizado con permiso del autor, el Dr. Antonio Aguilera. Sin embargo, para poder utilizar dicho método en este proyecto, se realizó una adaptación del mismo para que funcionara orientado a las percusiones, que son las señales que se manejan en este proyecto.

4.1.1 Arquitectura del sistema.

La arquitectura de esta aplicación es de tipo centralizada y es un sistema *"stand alone"*, es decir, que no necesitan o dependen de otras aplicaciones para funcionar. La arquitectura se describe a continuación en la figura 4.0.

Arquitectura del sistema

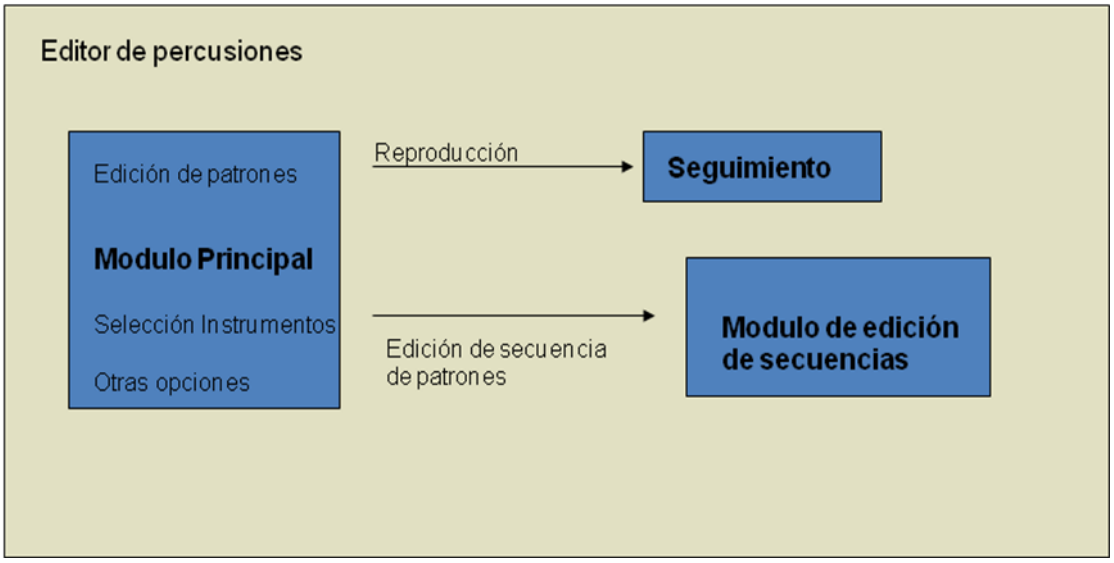

Fig. 4.0 Aplicación "stand-alone".

4.2 Método para tocar una percusión en Delphi.

Como se mencionó anteriormente, el método para tocar una percusión en Delphi se basa en un método que toca una nota genérica (ya sea de percusión o de algún otro instrumento) la cual es la siguiente:

Método genérico para tocar una nota de cualquier instrumento:

```
midiout ( command, data1, data2 ) ;
```

Esta instrucción llama al método "`midiout`", necesita tres parámetros como se especifica en los paréntesis. Este método sin embargo, a pesar de tener tres parámetros, necesita que se le especifiquen además, otros datos que aportan información importante para la ejecución del sonido que se escuchará.

4.2.1 Definición del primer parámetro del método "`midiout`".

El primer parámetro es muy importante ya que es un parámetro que podría llamársele compuesto. Para determinarlo, es preciso saber dos partes importantes de él. En el esquema 4.1 se pueden observar los datos adicionales que deben de ser especificados como parte de un parámetro en el método.

Esquema para determinar la información de los parámetros

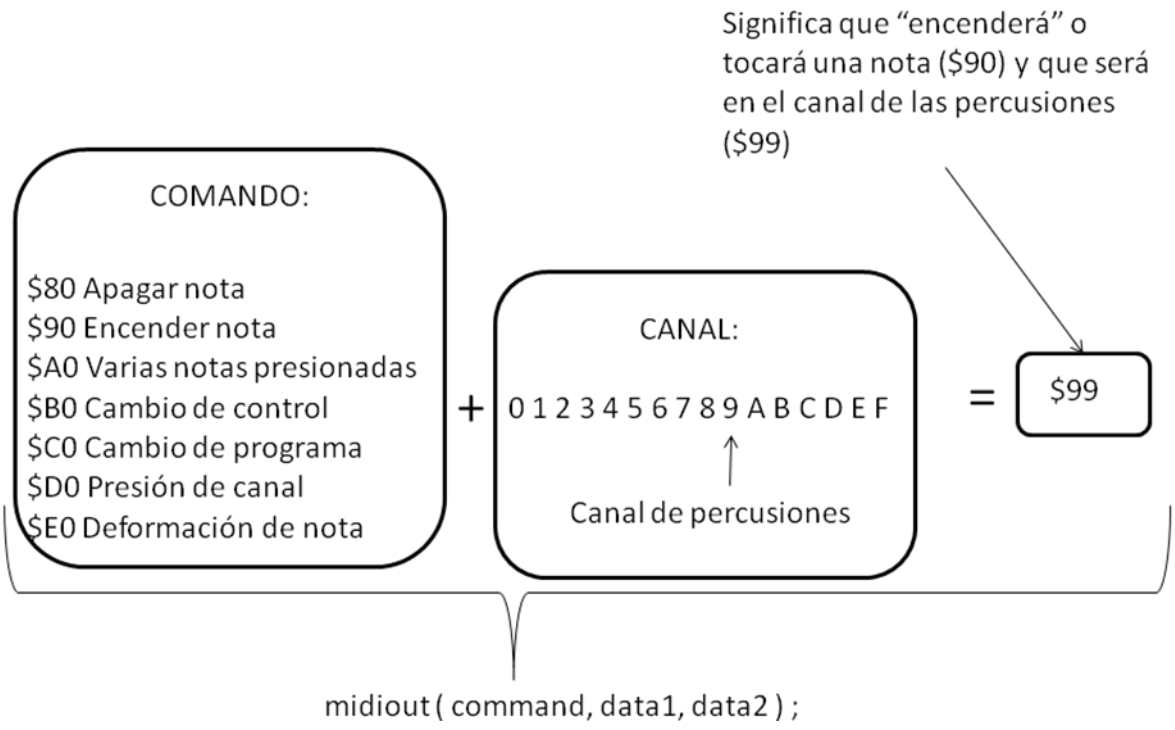

Fig4.1 Descripción de la estructura del parámetro "command".

De esta manera, tenemos que el parámetro para el método en este proyecto y que corresponde a *"command"* o comando para tocar una percusión, será de ahora en adelante $99 para todas las ocasiones en que necesitemos llamar al método.

4.2.2 Definición del segundo parámetro del método "`midiout`".

El segundo parámetro, se refiere al instrumento que se desea tocar, y éste va a depender del comando y del canal que se haya elegido. En este caso y para este proyecto en particular, como el comando y el canal son fijos, entonces se puede establecer que siempre se indicará el "encendido" de una nota y que será en el canal 9 de las percusiones, por lo que el segundo parámetro sólo constará del número o valor MIDI del instrumento que se desea tocar, es decir, uno de los valores descritos en la tabla 2.2 en el segundo capítulo. El esquema de este parámetro es el descrito a continuación en el esquema 4.2.

Esquema de funcionamiento del segundo parámetro.

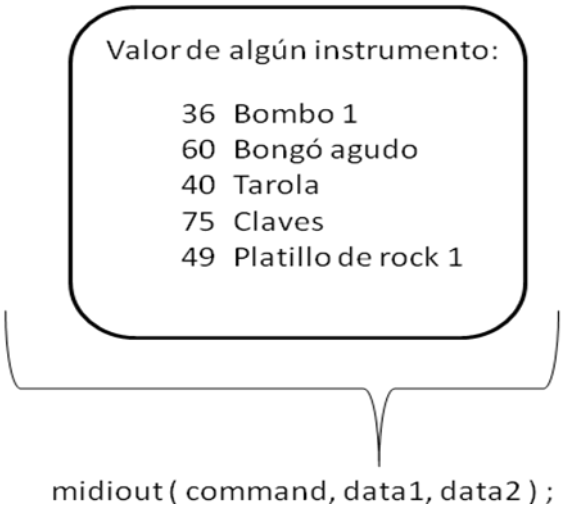

Fig. 4.2 Estructura de funcionamiento del segundo parámetro.

4.2.3 Definición del tercer parámetro del método "`midiout`".

El tercer parámetro se refiere al volumen con que se escuchará el instrumento al ser reproducido. El volumen en el estándar MIDI, consta de 128 niveles que van desde el 0 que es el nivel más bajo (silencio), hasta el 127 que es el nivel más alto. En éste parámetro

simplemente debe de ir especificado el nivel de volumen con el que debe de ser reproducida la nota. El esquema de funcionamiento del tercer parámetro es a como se describe a continuación en la figura 4.3.

Esquema de funcionamiento del tercer parámetro.

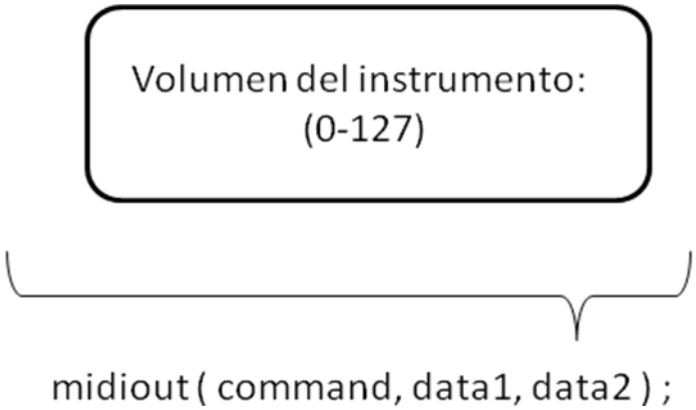

Fig. 4.3 El volumen en el parámetro puede tomar valores entre 0 y 127 incluyendo tanto al 0 como al 127 para todos los instrumentos.

Ya habiendo especificado las tres partes de los parámetros, se puede determinar la llamada al método que se utilizará en este proyecto. Por ejemplo si se desea golpear la tarola a un volumen de 100, la llamada al método que realiza la reproducción del golpe, se define a como se muestra a continuación:

```
midiout ( $99, 40, 100 ) ;
```

La razón del porqué esos valores en los parámetros de la llamada al método "`midiout`", se explican de la siguiente manera: $99 para indicar que se va a "encender" o tocar una nota en el canal 9 (que es el de las percusiones), el valor de 40 denota que es una tarola la que será golpeada y el valor de 100 indica el nivel de volumen con el que será escuchado.

4.2.4 Adaptación del método "`midiout`" a este proyecto.

En este proyecto, se utiliza la estructura del método "`midiout`" descrito en los puntos anteriores, sin embargo, se hizo una pequeña adaptación para hacer el llamado a éste de una manera general y cómoda para esta aplicación.

Si observamos bien la llamada en el método original descrita en el punto anterior (4.2.3), la llamada se hace de esta manera:

```
midiout ( $99, 40, 100 ) ;
```

Ahora bien, se estableció que el primer parámetro de esta llamada constaría de $99, el cual corresponde al de comando de $90 (encendido de nota) y canal 9 (de percusiones). Entonces, si se toma en cuenta que este parámetro siempre será el mismo para este proyecto, podemos establecer la creación de un nuevo método llamado *"play"*, que simplemente llame al método "`midiout`" con el primer parámetro como constante de la siguiente manera:

Método "*play*" adaptado para el proyecto.

```
procedure Play (Instrumento, Volume: Byte);  //Reducción a dos parámetros
//variables
begin
  midiout ($99 ,Instrumento, Volume) ; //Método original con un parámetro
constante
end;
```

4.3 Implementación de la selección de instrumentos.

Para llevar a cabo la implementación de la parte de selección de instrumentos, se tomaron los descritos previamente en la sección 2.2 del segundo capítulo y enlistados en la tabla 1.2 anteriormente. Se utilizó la herramienta llamada ComboBox la cual es una lista de donde el usuario puede seleccionar qué instrumento desea escuchar o agregar al área de edición. En ella, se colocaron los nombres de los 64 instrumentos en el orden que especifica el estándar de manera que a cada instrumento le corresponde un índice en la lista de cero a 64. Para hacer la asignación de valores MIDI simplemente se sumó 24 al índice donde se encontraba el nombre de cada instrumento en la lista del ComboBox y de esta manera, se obtiene el valor MIDI del instrumento. El área de selección de instrumentos en la aplicación es la que a continuación se presenta en la figura 4.3.

Área de selección en la aplicación.

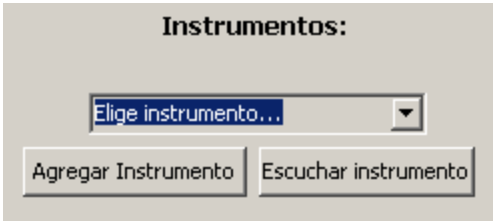

Fig. 4.3 En esta área, el usuario puede seleccionar del ComboBox, el instrumento deseado y agregarlo al área de edición o simplemente escuchar como suena.

4.4 Implementación del área de trabajo (edición de patrones).

Editar un patrón requiere principalmente de cuatro elementos. El primero de ellos es determinar qué instrumentos participarán o dejarán de hacerlo en la edición del patrón. El segundo es el de poder especificar en qué momento con respecto al tiempo deberán ser golpeados los instrumentos. El tercer elemento es la especificación de la resolución del patrón, en otras palabras, cuánto durará. El cuarto aspecto es determinar las variables que afectarán directamente a la reproducción del patrón como el volumen individual de cada golpe, el volumen en general del patrón y la velocidad.

Con estos cuatro elementos en conjunto podemos definir la edición básicamente de cualquier patrón que se nos pueda ocurrir. A continuación se detallará la implementación de estos cuatro elementos de edición.

4.4.1 Implementación del área de edición.

La edición de los patrones se lleva a cabo con un elemento llamado StringGrid. Este elemento es una cuadrícula en donde existen columnas y renglones, dichas columnas y renglones están clasificados en dos tipos, las fijas y las editables. En las columnas y renglones fijos, se encuentran elementos que el editor asigna como nombres de instrumentos, número de tiempo o *"beat"* y número de instrumento. La imagen 4.4 ilustra la cuadrícula que se encuentra en la interfaz del editor.

Cuadrícula de edición del editor

#	Instrumento	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	Bombo																
2	Tarola																
3	Hi-hat																
4	Crash																
5	Ride																

Fig. 4.4 La cuadrícula contiene propiedades editables (en blanco) y no editables (en gris).

Dentro de ésta cuadrícula, se puede especificar qué instrumento se golpeará y con qué intensidad, mediante la introducción de un valor numérico entero que puede ir desde el 0 (el cual significa eliminar el golpe) hasta el 9, valor máximo de volumen de golpe para los instrumentos. De manera que la cuadrícula se llena con números en los lugares donde se desea que el editor reproduzca el sonido de un golpe a un instrumento determinado. La cuadrícula llena se puede apreciar a continuación en la Imagen 4.5.

Cuadrícula de edición

#	Instrumento	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	Bombo	5				5	5			5				5	5		
2	Tarola			4				4				4				4	
3	Hi-hat	4		4		4		4		4		4		4		4	
4	Ride		3		3		3		3		3		3		3		3

Fig. 4.5 La cuadrícula con los golpes que deseamos que lleve a cabo y con su valor numérico del volumen con el que queremos que sean golpeados al momento de su reproducción.

En esta parte de la implementación es preciso mencionar que las propiedades de la cuadrícula son editables en el ambiente de desarrollo. La cuadrícula tiene propiedades de columnas fijas y no fijas, así como de renglones fijos y no fijos. Las propiedades iniciales con las que cuenta la cuadrícula es de 16 columnas editables, 5 columnas fijas de las cuales 3 no son visibles al usuario, en ellas se encuentra información que más adelante se describirá, también se tienen 6 filas iniciales, una de ellas es fija y actúa como renglón donde van los títulos de las columnas, las demás filas son editables.

4.4.2 Instrumentos en el área de edición.

Para especificar qué instrumentos se desean agregar al área de reproducción, se debe de considerar lo explicado en la sección 4.3. Una vez que se ha hecho la selección del instrumento a agregar, se necesita incluir la información de su valor MIDI y el nombre dentro de la cuadrícula que es el espacio de edición donde se manipulará la manera de sonar del instrumento, por lo que la manera de proceder es la siguiente: Se agrega un renglón a la cuadrícula conteniendo el nombre del instrumento en una celda visible pero no editable y la información del valor MIDI en una celda oculta y no editable para ser manipulada, las demás celdas editables y visibles, servirán para que el usuario especifique los golpes y el volumen de cada uno de ellos.

El proceso contrario es el de eliminar un instrumento del área de edición. Para ello es necesario considerar dos escenarios: el primero es si el instrumento a eliminar se encuentra en la última fila o si es una fila intermedia. En el primer escenario simplemente se remueven los datos de la última fila y se elimina la misma de la cuadrícula. En el segundo escenario se necesitan eliminar los datos únicamente y recorrer las filas subsecuentes hacia arriba, quedando la última línea vacía y lista para que sea eliminada.

Este procedimiento se hace en la interfaz mediante la selección del renglón en donde se encuentra la fila del instrumento que se necesita eliminar. A continuación en la figura 4.6 se muestran los elementos gráficos de la aplicación que realizan esta tarea.

Elementos para eliminar instrumentos de la cuadrícula.

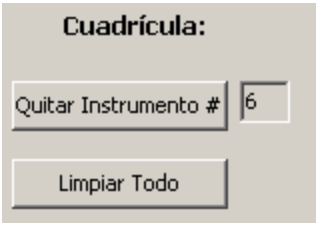

Fig. 4.6 Elementos de la interfaz que realizan la eliminación de los instrumentos en la cuadrícula.

4.4.3 Especificar la resolución del patrón en el área de edición.

La resolución del patrón dependerá del tamaño que se desee que tenga. En la cuadrícula se puede expresar su resolución por tiempos o *"beats"*. Estos tiempos pueden ser tantos como se deseen. El editor maneja 16 tiempos como valor inicial, también existe una opción de cambiar a una resolución de 32 tiempos al palomear un CheckBox (ya que estas dos resoluciones son típicamente, las más utilizadas), sin embargo se puede especificarle al editor la resolución que se desea que el patrón posea escribiendo el valor entero de la resolución deseada.

En cualquiera de estas situaciones, para realizar la conversión de resolución en la cuadrícula, simplemente se aumentan el número de columnas de la misma y se numeran con respecto del número de tiempos que se desea tener. Se muestra a continuación en la figura 4.7 la parte en la interfaz que realiza estas funciones.

Conversión de resolución del patrón expresado en la cuadrícula

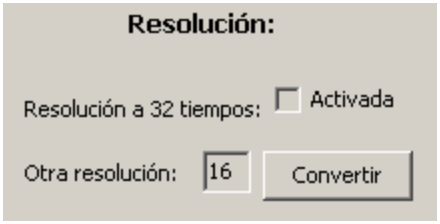

Fig. 4.7 En esta parte de la interfaz se puede especificar la resolución que se desea.

4.4.4 Variables generales de reproducción.

Dichas variables son en concreto 3, el primero es el volumen, el segundo es el estado de la reproducción (en progreso o detenido) y el tercero es la velocidad de reproducción. A continuación se detallan cada uno de estos elementos.

4.4.4.1 Primera variable, el volumen general de la aplicación.

El volumen tiene que ver directamente con el llamado al método adaptado de "`midiout`" para este proyecto llamado *"`play`"*, el cual recibe dos parámetros, el valor del instrumento MIDI y el volumen con el que se va a tocar (ver sección 4.2.4). El volumen puede definirlo el usuario en la interfaz mediante un elemento ScrollBar, una barra de desplazamiento donde pueden especificarse los valores de volumen global de la aplicación que van de 0 a 127.

4.4.4.2 Segunda variable, el estado de reproducción.

El segundo elemento, que es el estado de la reproducción tiene dos estados, detenido y en progreso. Dichos estados son controlados por el elemento mencionado anteriormente en la sección 4.1, el TTimer. Éste elemento, es el encargado de llevar ciertas acciones cada determinado intervalo de tiempo cuando el estado de reproducción esta en progreso. No lleva a cabo ninguna acción más que las de inicialización cuando el estado de la reproducción esta en detenido.Cuando el estado de la reproducción esta en progreso se llevan a cabo una serie de acciones. La principal acción que se lleva a cabo cada cierto intervalo de tiempo (o velocidad de reproducción) es verificar en la cuadrícula qué instrumentos son los que se van a tocar en ese tiempo. La manera de llevar a cabo esto es verificando columna por columna si las celdas editables de la cuadrícula tienen o no un número y si lo tienen, identificar de qué número se trata, ya que ése será el nivel de volumen del golpe para el instrumento en ese tiempo. Dentro de las acciones que se llevan a cabo en el TTimer se encuentran las siguientes, que son la base de la reproducción de la aplicación.

Acciones que lleva el elemento TTimer cada cierto intervalo de tiempo.

```
//si no está en modo editando (poniendo algun carácter en el grid)
if not (editando) then
//se selecciona el beat en el que vaya
 StringGrid1.Selection := TGridRect(Rect(beat,0,beat,19));
 for j := 1 to StringGrid1.RowCount-1 do//verifica en todas los renglones
//si tiene algo escrito la celda
        if Length(StringGrid1.Cells[beat,j])>0 then
        begin
//toma un solo caracter de la celda en el beat actual
            c := StringGrid1.Cells[beat,j][1];
            if c in ['1'..'9'] then//si es un numero de uno a nueve
            begin
//toma el numero que está escrito, lo convierte a entero y lo guarda en v
                v := StrToInt(c);
//toca la nota considerando el nivel de golpe (nueve niveles)
                play(Instrumento, Volumen* v div 9);
End
```

4.4.4.2 Tercera variable, velocidad de reproducción.

La tercera variable que se consideró para reproducir un patrón, es la velocidad. Dicha variable también puede ser manipulada por el usuario utilizando un elemento de tipo ScrollBar, el cual también es una barra de desplazamiento donde se puede especificar la velocidad con la que se desea que se reproduzca el patrón que se construyó. En cuanto a su representación existen dos aspectos que debemos de considerar. El primero de ellos es explicar cómo se maneja el tiempo internamente en la aplicación. El manejo del tiempo, como ya se había mencionado, lo lleva un elemento de Delphi llamado TTimer, el cual es un temporizador, que lleva a cabo ciertas acciones cada cierto período de tiempo. Este período puede ser especificado y establecido en una posición fija o puede ser manipulado si se desea. Como se pretende que el usuario tenga el control sobre la velocidad de la reproducción, se implementó el manejo del intervalo de respuesta mediante la barra de desplazamiento mencionada. La barra de desplazamiento, indica cuantos ciclos por minuto se llevarán a cabo. La aplicación tiene como valor predeterminado iniciar con 60 ciclos por minuto. Para convertir esto a intervalos expresados en milisegundos (ya que así lo maneja el objeto TTimer) debemos de dividir el número total de milisegundos en un minuto (60000 miliseg por minuto) entre la duración de un ciclo (4 golpes por ciclo) entonces se tiene que cada golpe en el ciclo constara de 15000 milisegundos (esto para un ciclo por minuto). Entonces intervalo de acción del TTimer estará dado por:

Intervalo de acción del TTimer (velocidad) = 15000 / ciclos por minuto deseados (1-250)

4.4.5 Implementación de la variación aleatoria de cambio de volumen.

La implementación de esta parte del proyecto consiste en considerar los tres aspectos mencionados en el diseño de dicha parte mencionados en la sección (3.4.1). Para poder modelar esta parte, se consideran los nueve niveles de intensidad de golpe con que se toca una percusión en un tiempo determinado, si ese valor que va de 1-9 es considerado como un porcentaje por ejemplo 1/9, 2/9, 3/9… 9/9, entonces al multiplicarse por el valor general de la aplicación se estaría indicando, dependiendo del nivel de volumen de golpe, qué porcentaje del volumen general se desea para ese golpe en particular. Esto determinaría el volumen de una manera estándar.

Para poder implementar la variación aleatoria de cambio de volumen se necesita además incluir un rango aleatorio. En esta ocasión, se definió que la variación sería de +/-40 niveles MIDI. Una vez definidos estos elementos, podemos tomar en cuenta ahora, el factor de cambio de volumen, el cual será un porcentaje que indicará en qué medida se hará notorio este cambio.Para lograrlo se deben obtener las ecuaciones para la variación aleatoria de cambio de volumen, conformada por la variación y la formula de volumen final:

Cálculo de la variación de volumen.

Variación= (-factorVolumen * porcentajeVariación) + random*

(factorVolumen*porcentajeVariación + factorVolumen*porcentajeVariacion)

Donde: random = un número aleatorio generado en tiempo de ejecución que va de 0 a 1.

Fórmula de volumen final:

Volumen final= VolumenGeneral*(VolumenDeGolpe/9)+Variación

4.4.6 Implementación del cambio aleatorio de instrumentos compatibles.

La implementación de esta característica se hace mediante el uso de una matriz de equivalencias. En ella, se utilizó el índice del número de columna para designar a cada uno de los grupos descritos en la tabla 3.5 en la sección 3.4.2 (9 grupos). En la matriz se colocaron los valores MIDI de los instrumentos que pertenecen al grupo, cada grupo en su respectiva columna y la columna cero se utilizó para guardar el número de instrumentos contenidos de cada grupo. La matriz de equivalencias queda entonces de la manera siguiente, descrita por la figura 4.8.

Estructura de la matriz de equivalencias

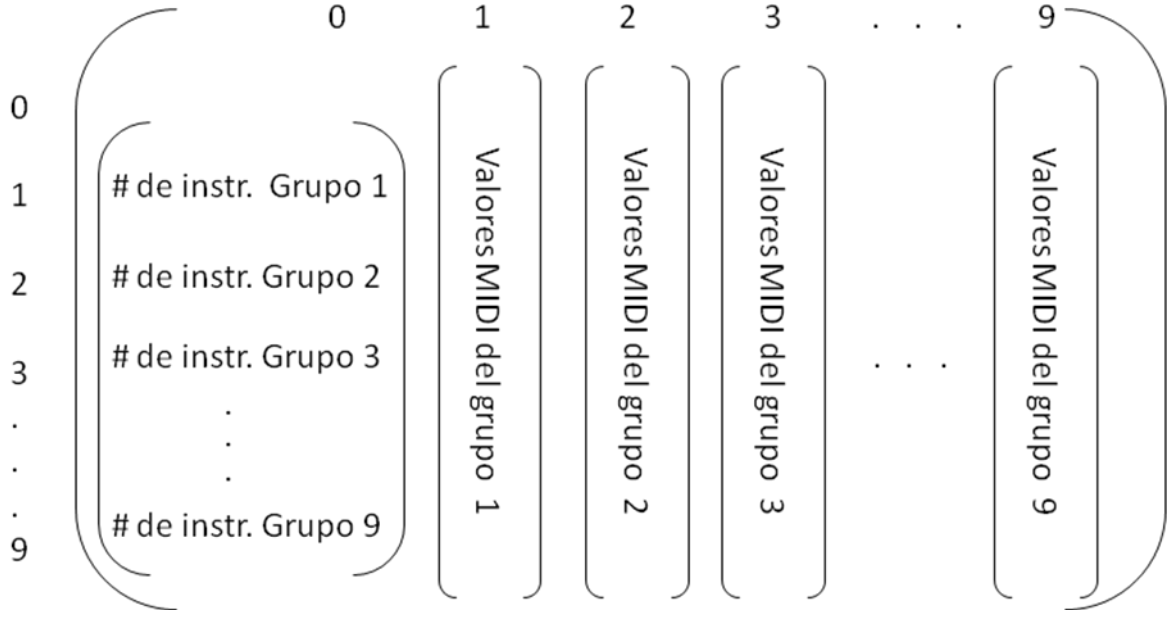

Fig. 4.8 La columna cero de la matriz, contiene el número de instrumentos de cada grupo.

Ya que se tiene esta matriz, se necesita saber a qué grupo pertenece un determinado instrumento a la hora de hacer el cambio por un instrumento compatible. Para ello, se utiliza un arreglo simple de enteros, el índice se utiliza para representar a cada instrumento, después, a la posición del arreglo se le asignan los valores de los grupos a los que pertenece el instrumento representado por el valor del índice. El arreglo de pertenencias tiene la estructura que muestra la figura 4.9 a continuación.

Arreglo de pertenencias

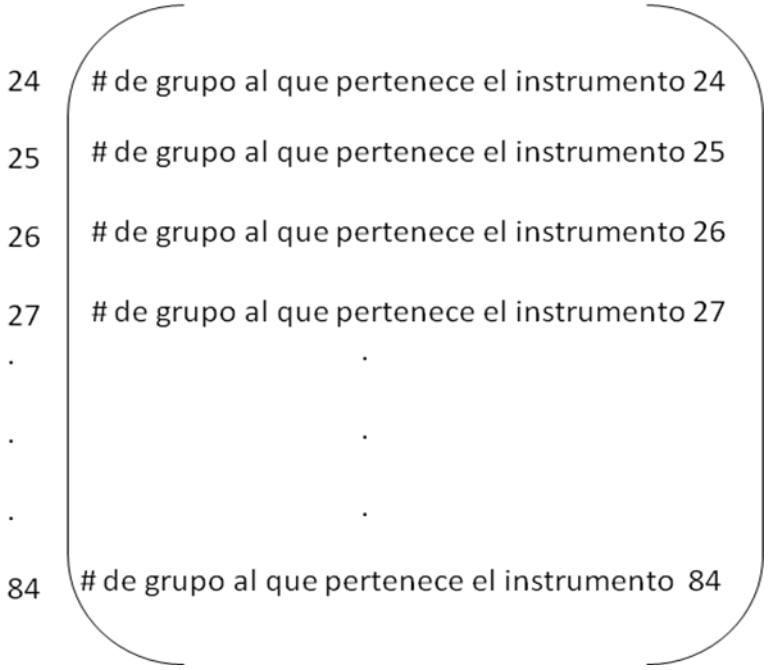

Fig.4.9 El índice es el valor MIDI del instrumento al que representa ese índice, el arreglo tiene asignado en esa posición, el valor del grupo al que pertenece el instrumento.

Ahora que se cuenta con la estructura que mapea los instrumentos a su grupo de pertenencia y que se puede determinar cuáles son los instrumentos que son compatibles, podemos realizar la llamada a un instrumento compatible en razón de su factor de cambio de instrumentos.

Se asigna a una variable un valor aleatorio entre cero y uno, si éste valor es menor al porcentaje de cambio de instrumentos, entonces se lleva a cabo la sustitución de instrumentos por uno que sea compatible. El procedimiento para llevar a cabo esto es el siguiente:

```
Randomize;
     i:=Random;
     if i<=FactorCambio then          //si esta dentro del porcentaje toca
un equivalente
           begin
                     v := StrToInt(c);
                      instrumentodelgrid:=
strtoint(StringGrid1.cells[0,j]);//en esta variable se guarda el valor
del instrumento actual

// se toca un instrumento aleatorio dentro del grupo de instrumentos
equivalentes

      play(equivalente[pertenece[instrumentodelgrid],Random(equivalente[0
      ,pertenece[instrumentodelgrid]])],Volume*v div 9+randVolumeint);
          end
```

4.5 Archivos de patrón.

Una característica importante después de haber compuesto un patrón, es poder guardarlo en un archivo para su posterior recuperación. En los puntos siguientes, se explica cómo se implementaron estas dos funciones tanto para guardar los archivos de patrón como para recuperarlos.

4.5.1 Guardar un archivo de patrón

Una vez que se ha compuesto un patrón, podemos guardarlo en un archivo. El editor, cuenta con esta funcionalidad y para la implementación de la misma se procedió de la siguiente manera.

Se debe de tomar en cuenta, que la composición del patrón consta tanto de su estructura, como de la especificación de las características de reproducción del mismo. Debido a ello, se necesitan guardar las características de reproducción del patrón que se construyó, así como su resolución y su estructura. Toda esta información está contenida de manera visible y no visible en las celdas de la cuadrícula o espacio de trabajo. De esta manera, el procedimiento a realizar es el de guardar toda esta información en un archivo de texto, ya que al hacer esto, podemos guardar los valores que poseía el patrón al momento de terminar de ser editado. Esto se implementó utilizando el método SaveToFile del objeto TStringList, ya que éste objeto tiene la propiedad de poder recorrer línea por línea de la cuadrícula, mapearla línea por línea como un String y poder guardarla a un archivo. Se especifica la ruta y nombre del archivo mediante un cuadro de diálogo y se le especifica la extensión .pat (abreviación de "patrón"). El método que lleva a cabo esta función es el siguiente:

```
begin
  with TStringList.Create do
    try
      if SaveDialog1.Execute then              // cuadro de diálogo para
salvar archivo
        begin
          for i:= 0 to StringGrid1.RowCount - 1 do
            Add(StringGrid1.Rows[i].CommaText); //se guarda en el archivo
el texto que hay en el grid
            temp:= SaveDialog1.FileName;
            SaveToFile(temp);
        end;
```

4.5.2 Recuperar un patrón desde un archivo.

Después de haber guardado el archivo de patrón conteniendo la información necesaria para reconstruirlo, se debe de recuperar dicha información para poder reproducirlo de nuevo. El primer paso es leer el archivo de patrón que se creó. En él, se encuentra la información de la estructura del patrón así como de las propiedades de reproducción del mismo.

Para recuperar un archivo de patrón, se especifica su nombre mediante un cuadro de diálogo. Sin embargo, no se debe de olvidar que además de recuperar la información del patrón, debemos de llevar a cabo ciertas acciones para colocar de nuevo al editor en las condiciones de reproducción en las que se encontraba cuando se guardó el patrón. Por ejemplo, una vez recuperado el valor del volumen, se deben de llevar las acciones necesarias para colocar la posición actual del volumen del reproductor en la que indica el archivo. Para implementar esto, se llevó a cabo el proceso inverso al de guardar un archivo, con la diferencia de que al terminar de recuperar la información de la estructura del patrón y ponerla en la cuadrícula, se llevan a cabo una serie de asignaciones de valores de reproducción para colocar el volumen y la velocidad en las posiciones y valores originales de cuando se hizo el guardado del archivo de patrón. La implementación de estos procedimientos es la siguiente:

```
if OpenDialog1.execute then     //se se da click en abrir archivo
        begin
          temp:=OpenDialog1.FileName;  //seleccion del archivo a abrir

          i := length(temp);      //se cuentan las letras que contiene el
nombre y ruta del archivo
          repeat
            dec(i);
          until temp[i]='\';   //esto se hace para poner en la ventana el
nombre del archivo abierto

          LoadFromFile(temp); //se carga el archivo seleccionado desde el
cuadro de dialogo de abrir archivo
          Caption:='.::Percussion   Editor::.   -   '+   Copy(temp,i+1,
length(temp)); //se pone el nombre del archivo junto al nombre de la
aplicación
          StringGrid1.RowCount:=  Count;  //se  cuentan  las  líneas  que
contenía el archivo guardado
          for i:= 0 to Count - 1 do
            begin
              if(StringGrid1.Cells[0,0]='1')then  //si esa celda tiene un
uno
                begin
                  Checkbox1.Checked:=true;//quiere  decir  que  la  casilla
de 32 tiempos estaba activada
                end
              else
                begin      //recuperan el tamaño del grid original que se
guardó

StringGrid1.ColCount:=strtoint(StringGrid1.Cells[0,0])+StringGrid1.FixedC
ols;
                  StringGrid1.Rows[0].CommaText:= Strings[0];
                  Edit2.Text:=StringGrid1.Cells[0,0];
                  Button6.Click;
                end;
              StringGrid1.Rows[i].CommaText:= Strings[i];   //se recupera
el texto que había originalmente
            end;       //a partir de aquí se asignan los valores globales
que estaban originalmente, tiempo, volumen
            Timer1.Interval:=strtoint(Stringgrid1.Cells[3,0]);
            Scrollbar1.Position:=strtoint(StringGrid1.Cells[3,1]);
            Label1.Caption:='bpm: '+inttostr(Scrollbar1.Position);
            Scrollbar3.Position:=strtoint(StringGrid1.Cells[4,0]);
            Label3.Caption:='Volumen: '+inttostr(Scrollbar3.Position);
            editando:=false;
            beat:=5;
            edit1.Text:=inttostr(StringGrid1.RowCount);
        end;
```

4.6 Implementación del módulo editor de secuencias

La última parte de implementación de este proyecto es el módulo editor de secuencias de patrones. Éste módulo basa su funcionamiento en el supuesto de que existen archivos de patrón previamente guardados. Además, se requiere para su funcionamiento, que el usuario sepa de qué manera desea reproducir el patrón antes de sustituirlo por el siguiente (como veces de repetición y velocidad de cada patrón), ya que esto determinará la duración de reproducción de un patrón antes de cambiar patrón siguiente, es preferible que se cuente con esta información antes de comenzar a reproducir los patrones, esto con el objeto de no tener que hacer cambios una vez que se ha iniciado el proceso de reproducción.

Sabiendo estos datos, se puede comenzar a editar la secuencia de patrones que se desea reproducir en base a los archivos creados previamente. El primer paso es seleccionar qué archivo de patrón formará parte de la secuencia. Para llevar a cabo esta acción se siguen los siguientes pasos:

1.- Se elige un archivo de patrón mediante un cuadro de diálogo en donde el usuario puede buscar su archivo de patrón.

2.- Una vez elegido el archivo, se puede decidir entre cargar dicho archivo a la lista de patrones a reproducir o elegir otro patrón para agregarlo a la lista.

3.- El archivo agregado se puede visualizar tanto en la lista del editor de secuencias, como en la lista en la ventana principal al momento de la reproducción, también se sombrea el nombre del patrón actual en reproducción en la lista principal para su mejor visualización.

Estos tres pasos se tienen que seguir para cada archivo de patrón que se desea reproducir en la aplicación.

El editor de secuencias cuenta con varias partes de implementación para su funcionamiento. La primera parte es la de la elección de archivos a cargar, la segunda es la carga misma de los archivos y la tercera es el manejo de la lista de reproducción.

Para la elección del archivo a agregar, únicamente se enseña un cuadro de diálogo en donde el usuario busca el archivo de patrón a agregar. En cuanto a la segunda parte, una vez teniendo el archivo elegido, se carga el nombre del archivo a las listas en las dos ventanas, y se llena un `TStringGrid` (no visible al usuario), con la información de instrumentos y de variables de reproducción que son necesarios para reproducir el patrón. Esto se lleva a cabo de la siguiente manera:

```
begin
  if(Edit1.Text='')then
    begin
      application.MessageBox('Se requiere ingresar un número de
repeticiones','Acción requerida' );
    end
  else
    begin
      verificador := Edit1.text[1];
      if verificador in ['0'..'9'] then
        begin
          veces:=strtoint(edit1.Text);
   if (Edit2.Text='') then //Si no se ha seleccionado el archivo a cargar
            begin
              application.MessageBox('Se debe de seleccionar un archivo
de patrón','Acción requerida');
            end
          else
            begin
              for j := 1 to veces do
                begin
 Listbox1.Items.Add(Edit2.Text);//se pone en la lista de archivos a tocar
                end;
    Edit1.text:='1';
    Edit2.clear;//se limpia el campo para mostrar el archivo seleccionado
            end;
        end
      else
        begin
          application.MessageBox('Sólo se admiten números en esta
casilla.','Información');
        end;
    end;
end;
```

El manejo de la lista de reproducción, también está sub-dividido en dos partes, la lista de reproducción del editor de secuencias, y la lista de reproducción de la ventana principal en el editor de patrones. Cada una, posee un comportamiento diferente, debido a que la naturaleza misma de cada lista sirve a propósitos distintos. La lista que se encuentra en el editor de secuencias, se utiliza para especificar qué patrones conformarán la secuencia que se va a reproducir. En esta lista, se puede especificar el orden y la existencia de los patrones que se desean reproducir. También, si se desea, se puede eliminar alguno de ellos. Una vez definidos el orden y el número de veces de repetición de los archivos de patrón, se carga la lista y entonces, se está listo para reproducir. Se mantiene una variable apuntador, que indica qué archivo de patrón es el siguiente a tocar y también otra variable apuntador que nos ayuda a saber en qué posición se cargará el siguiente archivo de patrón en caso de ser necesario. Estas acciones se llevan a cabo de la siguiente manera en la implementación:

```
begin
archivocargado:=0;
apuntador:=0;
tocar:=0;
secargoarchivo:=false;
Form1.ListBox3.Clear;

for k := 0 to 50 do
  begin
    for l := 0 to 50 do
      Siguiente[k].Rows[l].Clear;
  end;

if(listbox1.Count=-1) or(listbox1.count=0) then  //si no hay elementos en
la lista
  begin
    application.MessageBox('Se necesita agregar archivos a la lista
primero','Acción requerida');//se informa que se tienen que agregar
primero
  end

else
  if(listbox1.count>0) then //si hay elementos en la lista
    begin
      for j := 0 to Listbox1.count-1 do     //se carga cada archivo al
arreglo invisible de string grids
       begin
```

```
          i := length(listbox1.Items.Strings[j]);       //se cuentan las
letras que contiene el nombre y ruta del archivo
          repeat
            dec(i);
          until listbox1.Items.Strings[j][i]='\';    //esto se hace para
poner en la lista de la ventana principal el nombre del archivo que se
reproducirá
          Form1.ListBox3.Items.Add(Copy(listbox1.Items.Strings[j],i+1,
length(listbox1.Items.Strings[j])));  //se coloca en la lista de la
ventana principal el puro nombre
          Form1.Caption:='.::Percussion   Editor::.   -   Secuencia   de
múltiples patrones';//cambia el titulo de la ventana
          button1.Click;     //con este boton se hace el cargado del
archivo al arreglo invisible de StringGrid;
        end;
    end;

  if (secargoarchivo=false) and (listbox1.Count>0) then
    begin
    Button8.Click;
    Form1.Listbox3.Selected[0]:=true;
    inc(tocar);
    end;

    if(listbox1.Count>0)then
      begin
        application.MessageBox('¡Lista            cargada            con
éxito!','Información'); //se informa al usuario que la lista fué cargada
exitosamente
        Form1.BringToFront;
      end;
end;
```

Los botones uno y ocho en el código anterior, llevan a cabo las funciones de cargar un archivo y tocar un archivo respectivamente, sin embargo, son invisibles para el usuario y sirven propósitos internos a la aplicación. Ambos son parte importante en el manejo de las secuencias y para el cargado de archivos a la estructura que almacena la secuencia de reproducción. El botón uno, lleva a cabo procedimientos muy similares a los realizados cuando se abre un archivo de patrón desde el editor o ventana principal, la diferencia es que en vez de cargarlos a la cuadrícula principal en el editor para ser reproducidos, los carga en el arreglo donde se almacena la secuencia de patrones. El código es muy similar al de abrir archivo, como a continuación se puede apreciar:

```
with TStringList.Create do
    try
      LoadFromFile(Listbox1.Items.Strings[archivocargado]); //se carga el
archivo de la lista
      siguiente[apuntador].RowCount:= Count; //se cuentan las líneas que
contenía el archivo guardado
      siguiente[apuntador].Rows[0].CommaText:= Strings[0];
      for i:= 0 to Count - 1 do
        begin
          if(siguiente[apuntador].Cells[0,0]='1')then    //si  esa  celda
tiene un uno
            begin

siguiente[apuntador].ColCount:=32+Form1.StringGrid1.FixedCols;//quiere
decir que la casilla de 32 tiempos estaba activada
            end
          else
            begin        //recuperan el tamaño del grid original que se
guardó

siguiente[apuntador].ColCount:=strtoint(siguiente[apuntador].Cells[0,0])+
siguiente[apuntador].FixedCols;
            end;
          siguiente[apuntador].Rows[i].CommaText:=   Strings[i];      //se
recupera el texto que había originalmente
        end;       //a partir de aquí se asignan los valores globales que
estaban originalmente, tiempo, volumen
      apuntador:=apuntador+1;
      archivocargado:=archivocargado+1;
    finally
      Free;
    end;
```

El botón ocho, se encarga de transferir el patrón almacenado en la posición que apunta la variable "`tocar`" en la estructura donde se almacena la secuencia, a la cuadrícula principal. Se utiliza cuando se ha terminado de reproducir un patrón y se desea cargar el patrón siguiente que está en la estructura. Esto se lleva a cabo mediante la asignación, tanto de las variables que son característicos de cada patrón, como de la información que está contenida para la reproducción del mismo. Esto se lleva a cabo de la manera siguiente:

```
Form1.StringGrid1.ColCount:=siguiente[tocar].ColCount;     //se ajusta el
grid principal al grid que contiene los datos siguientes
  Form1.StringGrid1.RowCount:=siguiente[tocar].RowCount;

  for i := 0 to siguiente[tocar].RowCount do
```

```
    begin
      Form1.StringGrid1.Rows[i]:=siguiente[tocar].Rows[i];     //se copian
los datos
    end;
  //se asignan a partir de aquí los valores globales de la reproducción.
  Form1.Timer1.Interval:=strtoint(Form1.Stringgrid1.Cells[3,0]);
  Form1.Scrollbar1.Position:=strtoint(Form1.StringGrid1.Cells[3,1]);
  Form1.Label1.Caption:='bpm: '+inttostr(Form1.Scrollbar1.Position);
  Form1.Scrollbar3.Position:=strtoint(Form1.StringGrid1.Cells[4,0]);
  Form1.Label3.Caption:='Volumen: '+inttostr(Form1.Scrollbar3.Position);
  Editando:=False;
```

Para el manejo de las secuencias, se necesita que en el "`TTimer`" se especifiquen las condiciones para llevar a cabo el cambio de patrón. Una de estas condiciones, es verificar si la variable que lleva el conteo de los tiempos, ha llegado al último de éstos, entonces se lleva a cabo el proceso de cargado del siguiente archivo de patrón a reproducir. La implementación para este procedimiento es el siguiente:

```
if (beat>=StringGrid1.ColCount) then //si se ha llegado al final del
numero de beats
       begin
         if (siguiente[tocar].Cells[0,0]='') then     //y no hay mas
patrones cargados
           begin
             if(checkbox2.Checked)  then   //y esta palomeada la casilla
de repetir patron
               begin
                 tocar:=0;
                 if              not(siguiente[tocar].cells[0,0]='')then
Form3.button8.click;
                 if(secargoarchivo=true)then Listbox3.Selected[0]:=true;
                 inc(tocar);
                 beat:=5;// se repite el patron que esta actualmente en
el grid
                 if(listbox3.Count>0)then Listbox3.Selected[0]:=true;
               end
             else      // si no, se detiene la reproducción
               begin
                 button4.Click; //detener
                 tocar:=0;
                 if              not(siguiente[tocar].cells[0,0]='')then
Form3.button8.click;
                 if(secargoarchivo=true)then Listbox3.Selected[0]:=true;
                 inc(tocar);
                 if(listbox3.Count>0)then Listbox3.Selected[0]:=true;
               end;
           end
```

```
          else // si es que existen más patrones cargados...
            begin
              Form3.Button8.Click;      //se toca el siguiente
              beat:=5;                 //desde el inicio
              inc(tocar);               //se incrementa el apuntador para
tocar el patrón que sigue
              Listbox3.Selected[tocar-1]:=true;     //se   selecciona   el
patrón actual en reproducción que está en la lista
            end;
      end;
```

La segunda lista de secuencia, se ubica en la ventana principal o editor de patrones y cumple funciones de seguimiento de secuencia. En ésta, se puede dar seguimiento a la reproducción de la secuencia establecida previamente en el editor. También tiene la característica de que al hacerle clic en alguno de los patrones que se encuentran en ella, reproduce la secuencia a partir de ese patrón. Esta funcionalidad se lleva a cabo asignando los valores apropiados a los apuntadores, los cuales son los encargados de guardar los valores que se necesitan para llevar a cabo el cargado del siguiente patrón en la lista. Esta funcionalidad se lleva a cabo asignando los siguientes valores en los apuntadores:

```
begin
  beat:=5;  //valor para reproducir desde el inicio el patrón.
  tocar:=Listbox3.ItemIndex;//Valor  que  apunta  al  siguiente  patrón  a
reproducir
  archivocargado:=Listbox3.ItemIndex+1;    //apuntador  para  la  siguiente
posición en la estructura que guarda los patrones
  apuntador:=Listbox3.ItemIndex+1;    //  apuntador  para  archivo  que  se
cargó
  Form3.Button8.Click;//  tocar  el  patrón  al  que  apuntan  las  variables
anteriores
  inc(tocar); //se incrementa el apuntador al siguiente patrón.
end;
```