Software de Procesamiento y reproducción de Señales de Audio en Tiempo Real

Rafael Peña Martínez José Iván Pacheco Moo

Asesor de proyecto: José Enrique Álvarez Estrada

5 de diciembre de 2010

Universidad del Caribe, Cancún, Quintana Roo, México

Resumen

En este articulo se explora una posibilidad de virtualizar un pedal de efectos para guitarra, en dos vertientes: la generación de los efectos de audio mediante técnicas de procesamiento digital de sonido; y la creación de una interfaz de usuario que controle los efectos mediante de la detección de la posición del pie. Los efectos de audio que se exploraron fueron delay, echo y chorus. Se instrumentó la captura de audio y los algoritmos en MatLab. La interfaz de usuario usa una cámara web para detectar la posición del pie mediante reconocimiento de sombras. Se discuten los logros alcanzados, los retos enfrentados, en especial por las restricciones de tiempo real, y el trabajo por venir.

1. Introducción

Desde que la ciencia hizo su aparición en la escena musical a principios del siglo XX, el manifiesto literario de Filippo Marinetti (1876-1944) dentro del movimiento futurista (1906- 1916) proclamaba la adopción de principios estéticos relacionados con el impacto de la electricidad en el arte [SIELO]. Desde entonces la música evolucionó de una manera exponencial; debido a que la aplicación de la tecnología abrió un mundo de posibilidades tonales y armónicas para los músicos ya que permitieron explorar frecuencias que hasta el momento sólo eran parte de la teoría musical.

Sin embargo, fue en los primeros años a partir de que la electricidad hizo su aparición en el arte cuando se suscitaron la mayor cantidad de avances tecnologicos y aplicaciones de la tecnología en la música, con la aparición de amplificadores, micrófonos, guitarras eléctricas y algunos dispositivos que permitían distorsionar o aplicar algún tipo de compresión o efectos (como el eco, delay, flanger etc.) a la señal proveniente de las guitarras (aunque existen también para voz y otros instrumentos.). Desde entonces esta teoría básica no ha sufrido modificaciones que puedan considerarse revolucionarias, ni tampoco la visión con la que se construyen estos dispositivos.

Podemos pensar que la tecnología en la música a llegado a un punto en el que todo está inventado; a demás de que aun en estos tiempos donde todo apunta a que lo verdaderamente funcional es la virtualización, debido a que se evitan gastos en hardware innecesarios y la mayoría de los músicos siguen inmersos entre circuitos electrónicos para poder dar un toque personal a sus melodías (aunque es de reconocerse que el desempeño es excelente); y tal vez no sea por renuencia a las nuevas tendencias, sino simplemente porque nosotros, los encargados de crear tecnología no hemos podido ofrecer nuevas alternativas de igual o mayores prestaciones.

Existen muy pocas soluciones tecnológicas en esta área, que aborden el problema desde nuestra perspectiva, es decir no hay muchas opciones en cuanto a software se refiere, debido a que el concepto es relativamente nuevo y no se ha desarrollado del todo. Las soluciones que existen como el Amplitube, Guitarfx, Guitar-rig, no son del todo funsiónales ya que la latencia que existe entre la captura y la reproducción del audio es considerable.

Software de procesamiento y reproducción de señales de audio en tiempo real. Es el nombre que le hemos dado a nuestro proyecto, que en palabras más simples trata de virtualizar un dispositivo generador de efectos, llamado de manera coloquial pedal cuando es un módulo individual, o pedal multiefectos cuando en un solo pedal se encuentran integrados dos o más módulos de efectos.

El concepto de virtualizar un pedal de efectos no es del todo nuevo, sin embargo no ha sido desarrollado mas allá de sólo crear un software que procese señales de audio, les aplique algún efecto y en algunos casos (la mayoría eficientemente) reproduzca esta nueva señal en tiempo real.

Lo que pretendemos realizar es un software que pueda procesar las señales y reproducirlas en tiempo real, utilizando algunas bibliotecas que ya han sido desarrolladas para distorsionar las señal, fomentando así la reusabilidad de código. Para ello es necesario conocer la teoría básica del procesamiento de señales desde un punto de vista matemático y computarizado para poder tener las bases teóricas y garantizar que el procesamiento de la señal se llevará a cabo eficientemente y no dejar nada al azar.

Los problemas que debemos abordar con mayor detalle son sin duda:

- La parte de captura y procesamiento de las señales (en este caso particular serán señales de audio), utilizaremos algunos cálculos matemáticos realizados mediante la FFT (transformada rápida de Fourier) para crear nuestros efectos de audio de los cuales ya existen algunos modelos matemáticos ; además deberemos tomar en cuenta la taza de muestreo para evitar el problema de aliasing y el error de cubanización(conocido como Signal to Noise Relation)[DSP]. y poder obtener una mejor calidad en el sonido procesado.

- EL otro punto, que representó el reto más complicado, consistió en lograr la reproducción en tiempo real de la señal procesada.

- Otro aspecto que abordaremos será el diseño de una interfaz que respete el concepto básico de funcionamiento de un pedal de efectos, para que el usuario se sienta lo más cómodo posible en el momento

de ejecutar alguna pieza musical en su instrumento preferido. Aunque cabe mencionar que el diseño de la interfaz no es lo innovador, sino la manera en la que el usuario puede manipularla. Y como mencionamos, el objetivo planteado es que quien use este sistema no tenga que dejar de ejecutar el instrumento para realizar los cambios de efectos, sino que lo pueda hacer como si utilizara un pedal convencional.
Este es un proyecto que surge básicamente de una necesidad personal, la cual seguramente se ve reflejada en muchos otros músicos, que al igual que nosotros no cuentan con los recursos económicos necesarios para adquirir un procesador de audio profesional, ya que su costo es elevado. Es por ello que hemos decidido aplicar los conocimientos adquiridos a lo largo de nuestra carrera, para aportar una opción viable a todos aquellos músicos que se encuentran o se han encontrado en una situación similar a la de nosotros y están en espera de nuevas alterativas. Y es debido a las necesidades y carencias ya mencionadas que hemos decidido crear un software que cumpla con las características básicas de un pedal de efectos, el cual se ve favorecido por las siguientes ventajas:

- Cumplir con las características funcionales de un pedal común.

- Distribución libre. Al ser de licencia libre, el código estará disponible para ser modificado, por cualquier persona.

- En comparación con el software que se encuentra en este momento en el mercado, integrará una interfaz que dará al usuario la sensación de estar utilizando un pedal de efectos convencional.
El termino de virtualización es antiguo se viene usando desde 1960, y ha sido aplicado a diferentes aspectos y ámbitos de la informática, desde sistemas computacionales, hasta capacidades o componentes individuales, Lo más importante en este tema de virtualización es la de ocultar detalles técnicos a través de la encapsulación.

Existen en el mercado diferentes opciones de programas que emulan procesadores de audio y amplificadores, como por ejemplo Amplitube, revalver, guitar FX, por mencionar algunos. Debido que se puede obtener cualquiera de ellos de forma gratuita, podemos compararlos, analizando sus ventajas y desventajas. Uno de los principales problemas que presentan estos programas es el retardo de la señal de salida ya procesada respecto a la señal de inicio: la latencia. Además que el

consumo de recursos de la computadora son altos.

Amplitube, es una buena opción para usuarios que cuentan con un ordenador poderoso en cuestiones de procesamiento y memoria, pues su interfaz es pesada y por este motivo su rendimiento disminuye de manera considerable en un ordenador sin las características necesarias. Además la interfaz resulta incomoda para utilizarse en una presentación en vivo y no ofrece ninguna opción para solucionar este inconveniente.

Por su parte Guitar-rig y GuitarFx no cumplen con la reproducción en tiempo real, y además presentan el defecto de que mientras no detectan ninguna señal procedente de la guitarra, producen ruidos que son bastante molestos para quien lo utiliza.

Nuestro software en comparación con estos sistemas, cuenta con una opción para que la manera en que se maneja la interfaz se apegue a la forma en que se maneja un pedal de efectos convencional, además de que el código fuente estará disponible para que cualquiera que así lo desee pueda modificarlo y mejorar, adaptar o ajustar los parámetros que al usuario convengan y de esta manera poder personalizar el sistema. Esto sin duda es una ventaja sobre nuestros competidores ya que el sistema se irá diversificando para poder ofrecer mayores opciones y mejorar las prestaciones de nuestro software.

En cuanto al manejo de la interfaz existen algunas formas de implementarlo, por ejemplo, utilizando las técnicas de Jonhy Lee (wiimote control de wii) la cual se puede convertir cualquier monitor (monitores crt, monitores lcd, proyecciones, etc) a una pantalla táctil. Sin embargo utilizar esta técnica es muy costoso debido a los requerimientos de hardware necesarios.

Ahora bien existe otra alternativa como el Webcam Whiteboard, la cual utiliza una cámara web para detectar la posición que ha sido presionada sobre un panel, el cual puede ser construido sobre cualquier superficie transparente. Esto se implementará usando una biblioteca de funciones TouchLib escrita bajo el lenguaje C. Esta librería sirve para interactuar con superficies multitouch.

Con la ayuda de esta biblioteca de funciones es posible construir una superficie multitouch la cual utilizaremos para simular un pedal de efectos.

2. Desarrollo

Para comprobar la factibilidad de los objetivos planteados contamos con varias opciones para desarrollar nuestro proyecto, como son:

- Matlab. Por las herramientas y características que pose para realizar cálculos matemáticos y de esta forma poder diseñar e implementar los efectos que hemos pensado crear para nuestro sistema. Además de que nos permitirá crear fácilmente la interfaz de usuario [GUIM].

- Linux. Por la facilidad que nos ofrece a la hora de manejar nuestro dispositivo de audio: ya que maneja este dispositivo como un archivo, lo cual facilita tanto la lectura como la escritura.

- El Small Basic. Por ser un lenguaje sencillo de manejar lo cual nos permite implementar el diseño que realizamos para crear nuestros efectos.

- Octave. Que es un lenguaje muy similar a Matlab con la diferencia de que es de software libre.

2.1. Captura y Reproducción de sonido

El procesamiento digital de audio es la fase más crítica del proyecto, ya que si no se consigue reproducir las señales a tiempo real, el esfuerzo realizado hasta este punto habrá sido totalmente infructuoso;

ya que es uno de los objetivos que necesitamos cubrir en su totalidad ya que de esto depende que su desempeño sea mejor que los software de edición de que pudieran existir en el mercado. Cabe hacer mención, que para cumplir con este objetivo, (– checar con el profenos estamos basando en la idea de que con la configuración de fábrica de la tarjeta de audio de nuestro ordenador es más que suficiente para poder lograr el tiempo real.)

Las investigaciones que se realizaron para cumplir con este objetivo fueron realizadas tanto en manuales de Matlab como en foros Web especializados en temas de procesamiento, captura y reproducción de audio, de los cuales adquirimos información de personas que

han tenido algún acercamiento o experiencia en temas similares a los de nuestro proyecto.

Las primeras pruebas realizadas para la captura de audio fueron hechas utilizando el bash de Linux, tomando en cuenta que el manejo de todos los dispositivos que se encuentran instalados en nuestro ordenador son vistos como archivos, es decir cada uno de los dispositivos tienen una dirección como las de un archivo regular. Por ejemplo la ruta para localizar la tarjeta de audio es /dev/audio/"[AUDIO]. Sabiendo esto, ocupamos un script nativo que se encuentra instalado en la configuración típica de la mayoría de las distribuciones llamado .ªrecord.ªl cual se le deben pasar como mínimo la frecuencia de muestreo para que pueda realizar la captura del audio y la ruta en la cual se almacenara dicha grabación.

Ya programando en Matlab, realizamos varias pruebas con las siguientes opciones:

- analoginput('winsound'), analogoutput('winsound'). Son funciones de Matlab que se incluye en MATLAB Data Acquisition Toolbox, las cual nos permite obtener y escribir el audio directo de y hacia la tarjeta de audio, de forma analógica; de manera que al capturar y reproducir el sonido podemos darle las características que nosotros deseamos con la función "set", como por ejemplo la frecuencia de muestreo, los bits por cada muestra etc [ADA].

- msound. El cual está implementado basado en PortAudio, el cual nos permite interactuar de manera mas cómoda con la tarjeta de audio ya que está pensada para trabajar en tiempo real (aunque en nuestra experiencia no es del todo cierto ya que existe latencia en el procesamiento). Esta fusión tiene diferentes parámetros los cuales nos permiten, abrir, cerrar, escribir y leer de la tarjeta de sonido. [MSA].

Sin embargo la opción por la cual optamos fue la de utilizar la función "msoud"debido a que nos resultó mucho mas fácil la captura y reproducción del sonido, a demás de que nos permite jugar con los parámetros de captura y reproducción y no es necesario configurar todos estos, ya que la configuración por defecto es más que suficiente para trabajar este modulo. Cabe mencionar que para poder utilizar esta función en Matlab es necesario utilizar una librería DLL, ya que no es una función nativa de Matlab.

Debemos tomar en cuenta que la señal que capturamos es analógica y decidimos realizar el muestreo con una frecuencia de 44100Hz para poder obtener una calidad de CD y utilizamos 16 bits por muestra, debido a que si utilizamos mas bits por muestra aumenta el retraso en la reproducción del sonido capturado. El parámetro que utilizamos para decirle a la función msound que obtenga las muestras de la tarjeta de audio es "getStream", y como parámetro extra podríamos pasarle una vez más la frecuencia de muestreo, sin embargo esto nos podría resultar contra producente ya que podría producirse un efecto de aliasing, que como ya comentamos, es el error que se obtiene al muestrear con una frecuencia inadecuada es decir una frecuencia muy alta, más del doble de la frecuencia de la señal original; por lo cual debemos tener especial cuidado en esta situación [DSP].

Las pruebas realizadas, hasta este punto, tambien fueron implementadas en Octave que como ya se menciono es el equivalente a Matlab en Linux. Las diferencias en las pruebas unicamente radican en las funcione nutilizada para escribir los datos en la tarjeta de audio, ya que en Octave no existe el comando "soundsc"sin embarog su equivalente es "sound.ªl cual se le pasa como parametro la variable que contien los datos y puede o no especificarse la frecuencia y/o el numero de bit por muestra.

Una vez que encontramos la solución al problema de la captura y reproducción de audio nos enfocamos en la creación de los efectos que se implementarán en nuestro sistema. Este proceso se describe en la siguiente sección.

2.2. Módulos de efectos

El procesamiento de señales es sin duda uno de los pasos más importantes que debemos abordar en nuestro proyecto, ya que básicamente, este proyecto está basado en el procesamiento de señales, en este caso, señales de audio. Si entendemos la teoría básica del comportamiento de las señales de audio y la conversión de señales de audio de analógicas a digitales y viceversa el proceso de codificacion en Matlab nos resultara más sencillo.

En esta sección, fuimos avanzando paso a paso ya que comenzamos a analizar diferentes opciones para crear los efectos adecuados para este proyecto, ya que necesitamos que el nivel de procesamiento de los efectos sean los menos exigentes para nuestro procesador y de esta

manera disminuir la carga al procesador y así poder lograr una latencia mínima.

Ahora bien después de analizar dentro de una gran gama de posibilidades los efectos que podríamos implementar, nos decidimos a implementar efectos, los cuales, irónicamente son producidos gracias al retraso de la señal original, dentro de un sistema FIR (Finit Impulse Response) [AEM] el cual nos proporciona las características necesarias para crear nuestros efectos dentro de un sistema estable

Un sistema basado en filtros FIR está basado en una ecuación en diferencias, como la que a continuación se expresa:

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) \tag{1}$$

La señal de entrada x(n) es alimentada por las señales que han sufrido un retardo mediante una suma ponderada con las señales de salida y(n). Este sistema de alimentación es conocido como un sistema no recursivo. La Transformada Z de la ecuación anterior es:

$$Y(z) = b_0X(n) + b_1z^-1X(z) + b_2z^-2X(z) \tag{2}$$

$$= X(z)(b_0 + b_1z^-1 + b_2z^-2) \tag{3}$$

y se resuelve por Y(z)/X(z) y se obtiene la función de transferencia:

$$H(z) = Y(z)/X(z) = b_0 + b_1z^-1 + b_2z^-2 \tag{4}$$

La siguiente figura muestra el diagrama de bloques de un sistema FIR convencional.

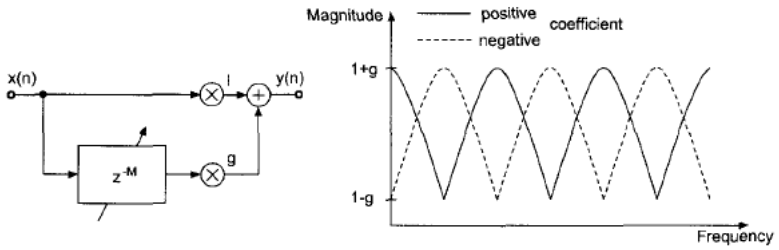

Figura 1: Diagrama de bloques de un filtro FIR convencional

Ahora bien los efectos que implementaremos son los siguientes:

- Dealy. El cual genera un retraso en la señal y se repite hasta que la fuente deje de generar una señal.

- Echo. El cual es una variante del delay, la diferencia es que la repetición de la señal no está sujeta al tiempo que tarde la fuente en dejar de generar una señal de audio; sino que repite la señal un número determinado de veces.

- Chorus. Este efecto también se desprende del delay. La diferencia es que se repite solo una vez y con un retraso controlado, que por lo regular es aproximado a 0.05 milisegundos. Este efecto es implementado par dar la sensación de que existen dos instrumentos ejecutándose a la par, es decir con una sincronía exacta.

A continuación explicaremos como es que cada uno de estos efectos está diseñado, bajo una base matemática, de igual forma se dará una explicación más amplia de lo que sucede con las señales de audio que son sometidas a este procesamiento.

2.2.1. Delay

Lo que le sucede a la señal de entrada con este efecto es que se retrasa por un período de tiempo determinado. El efecto es audible sólo cuando la señal procesada se combina (añade) a la señal de entrada, que actúa como una referencia. Este efecto, posee dos parámetros de ajuste: la cantidad de tiempo de retardo T y la amplitud relativa de la señal retardada a la señal de referencia [DFX]. La ecuación diferencial y la función de transferencia está dada por:

$$y(n) = x(n) + gx(n - M) \tag{5}$$

con

$$M = t/Fs \tag{6}$$

$$H(z) = 1 + gz^{-}1 \tag{7}$$

El tiempo de respuesta de este filtro se compone de la señal directa y el retraso en la señal original. Para los valores positivos de g, el filtro amplifica todas las frecuencias que son múltiplos de 1 / T y atenúa todas las frecuencias que se encuentran en medio. La función de transferencia del filtro muestra una serie de picos como se ve en la

figura anterior. Es por eso que este tipo de filtro se denomina filtro de pico. Para valores negativos de g, el filtro atenúa frecuencias que son múltiplos de 1 /T y amplifica las que se encuentran en medio. La ganancia varía entre 1 + g y 1 - g.

A continuación, mostraremos un ejemplo de la codificación de un filtro FIR que ejecuta un efecto de delay:

```
x=zeros(100,1);x(1)=1; - señal de pulsos unitaria de tamaño 100
Delayline=zeros(10,l); - asignación de memoria
for n=l:length(x);
y(n)=x(n)+g*Delayline(10);
Delayline=[x(n) ;Delayline(1: 10-1)] ;
end;
```

Así como los retrasos acústica, el filtrado tiene un efecto tanto en el tiempo dominio del tiempo como en el de la frecuencia. Nuestro oído es más sensible al efecto de uno u otro de acuerdo al rango tiempo en el que se establece la señalo. Para grandes valores T, se puede escuchar un eco que es distinto de la señal original. Las frecuencias que son amplificadas por el filtro son producidas tan cerca una de la otra, que apenas es posible identificar el efecto espectral. Para valores pequeños de T, nuestro oído no puede separar los acontecimientos en el dominio del tiempo por lo cual se puede notar el efecto espectral del filtro.

Ahora mostraremos el diagrama de bloques del Delay:

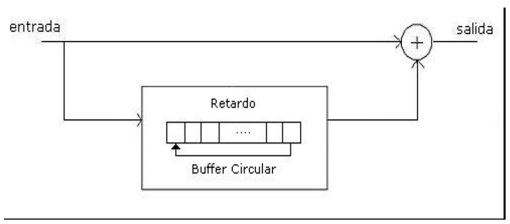

Figura 2: Diagrama de bloques

Ahora bien, como ya mencionamos con anterioridad, a demás del delay existen otros efectos que se desprenden de la misma teoría, es decir se basan en el retraso de la señal original y esto hace posible

poder producir un efecto de eco. Y es así como llegamos a la decisión de crear un par de efectos más basados en esta teoría, estos son el eco y el chorus, de los cuales ya describimos cual es el efecto que percibe el oído humano.

2.2.2. Echo

Este efecto consiste en la reproducción de la suma ponderada de muestras actuales con muestras anteriores. Las muestras antiguas son almacenadas en un buffer o memoria. En el momento de la reproducción la muestra antigua se multiplica por un factor de atenuación que disminuye su intensidad, lo cual da mayor realismo al efecto. El tamaño del buffer está relacionado con el retardo entre las muestras actuales y las antiguas. A esta técnica se le conoce como buffer circular ya que las muestras se almacenan desde la primera posición del buffer hasta la última, luego el contador del buffer se reinicia y por tanto comienza nuevamente el ciclo, esto se repite en un bucle infinito [DFX]. En resumen, a la muestra actual se le adiciona la muestra correspondiente a un retardo de:

$$T = longbuffer/Fs \tag{8}$$

donde Fs es la frecuencia de muestreo. Luego la muestra actual es almacenada en la misma posición de memoria donde se almacenó la muestra antigua. La figura 2 presenta diagrama de bloques del sistema que genera este efecto.

La siguiente imagen muestra el diagrama de bloques en el cual se describe como está diseñado este efecto.

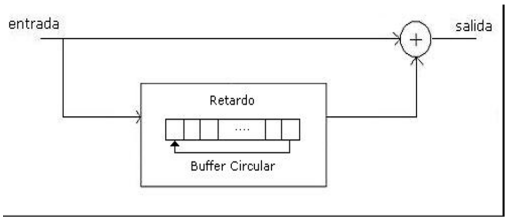

Figura 3: Diagrama de blouques del echo, es similar al delay, lo que cambia es el tamaño del buffer

2.2.3. Chourus

Este efecto consiste en simular un segundo instrumento en coro con el original. El principio es el siguiente: cuando dos personas tocan dos instrumentos juntos no lograrán una precisa sincronización, y al sumarse los dos instrumentos se detectarán algunos retardos entre un instrumento y el otro, los cuales serán variables. El chorus logra simular este efecto, a partir de un solo instrumento o señal de audio. El algoritmo consiste en copiar la entrada a la salida y sumarle a ésta una muestra retardada. El retardo usualmente utilizado es de entre 10ms y 20ms. La falta de sincronismo entre los dos instrumentos, no tendrá un retardo constante entre sí, si no que será variable, para ello se utiliza una señal periódica, como por ejemplo un seno que debe estar alrededor de los 3 Hz para obtener un buen resultado. La siguiente figura presenta diagrama de bloques del sistema que genera este efecto [DFX].

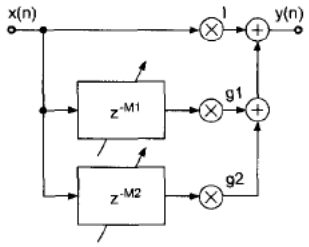

Figura 4: Diagrama de bloques del sistema chorus

Como hemos podido observar, estos efectos son básicamente variaciones en el tiempo en el que se reproducen las muestras de la señal de entrada; a continuación mostraremos una tabla en la cual se muestran los tiempos de retraso de la señal para algunos efectos, incluyendo el chorus y el echo:

Table 3.3 Typical delay-based effects.

Delay range (ms) (Typ.)	Modulation (Typ.)	Effect name
0 ... 20	-	Resonator
0 ... 15	Sinusoidal	Flanging
10 ... 25	Random	Chorus
25 ... 50	-	Slapback
> 50	-	Echo

Figura 5: Tabla de tiempos de retraso de los diferentes efectos diseñados

Después de conocer los factores que influyen, en el diseño de estos efectos, nos dimos a la tarea de codificarlos en Matlab, comenzamos por hacer las pruebas con archivos pregrabados, esto debido a que nos pareció mucho mas sesillo realizar las pruebas, debido a que no necesitamos la guitarra en todo momento y a demás de que es mas fácil conocer los parámetros de la señal de u archivo de audio que

ha sido grabado y digitalizado bajo un formato WAV en nuestro caso especial.

2.3. Interfaz

El diseño de la interfaz de usuario fue implementada con la herramienta GUIDE es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas .Tiene las características básicas de todos los programas visuales como Visual Basic o Visual C++.

Guide consta de dos archivos ".m"(ejecutable) y otro ". fig"la parte grafica. Las dos partes están unidas a través de las subrutinas callback. Una vez que se graba los archivos desde la consola de emisión (si salvamos la .fig automáticamente lo hace el .m asociado). El archivo .m que se crea tiene una estructura predeterminada. Consta de un encabezado y a continuación viene el código correspondiente a las siguientes subrutinas.

En la interfaz implementada se tiene 3 botones los cuales representarán los 3 a implementar. Cabe mencionar que existe una amplia gama de efectos, pero para fines prácticos se decidieron implementar 3 efectos básicos los cuales son el delay y el echo y el chorus las cuales serán llamadas con sus respectivas subrutinas con esta interfaz, se pretende mostrar un diseño de un pedal básico, así como imagen de fondo de nuestro diseño la cual tendrá la apariencia de un pedal real en la cual se insertaran los 3 botones de los efectos.

Una vez teniendo la interfaz grafica de usuario se decidió utilizar una herramienta OpenSource la cual es una biblioteca llamada touchlib que sirve para poder hacer que una cámara web pueda capturar las sombras de nuestros dedos o en este caso la sombra generada con el pie, en el cual la Cámara Web estará puesta en el fondo de pequeña caja y así pueda captura mejor las sombras generadas. En la parte superior de la pequeña caja colocaremos hoja blanca encima de la misma pondremos un pedazo de plástico o vidrio, esto nos servirá para poder tener una superficie solida y resistente y así simularemos un touchpad de mayor tamaño en el cual dicha sombra representara el movimiento de un cursor así como el click de un ratón .

Para la construcción del pedal, utilizaremos una caja que podría ser construida de cartón o sobre una base de acrílico la cual servirá para, asemejara nuestro pedal de efectos, en la cual le haremos un orificio en la parte lateral den el caso de que sea una caja para que pueda pasar

el cable USB de la cámara web y así poder ubicarla en el fondo de la caja y así pueda captar las sombras generadas pos nuestros dedos o en este caso con el pie y traducirlo en clics para controlar la interfaz de nuestro sistema.

En la siguiente figura obserbaremos el diseño final de la interfaz de usuario.

Figura 6: Tabla de tiempos de retraso de los diferentes efectos diseñados

3. Resultados

En cuanto a los resultados obtenidos podemos hablar de que las pruebas realizadas com Matlab y Octave no fueron del todo alentadoras, ya que en un principio supusimos que el diseño de nuestro softwa habiá sido erroneo ya que no lograbamos alcanzar el objetivo del timpo real, el retraso era considerable tan solo en la face de captura y reproducción, sin procesar ninguno de los efectos diseñados. Sin embargo, despues de un analisis detallado, acerca del diseño de uestr software nos dimos cuenta de que en realidad, el diseño del software en las etapas de procesamiento de efectos si lograba ejecutarse en un timpo considerablemente bajo, tanto en Matlab como en Octave. Esto lo pudimos constatar, midiendo el tiempo que transcurre, despues de haber obtenido una muestra de audio y hasta que es procesado el

efecto. Este tiempo de procesamiento es minimo, y podriamos desir que imperceptible al oido humano; los tiempos de respuesta son los siguietes:

- Matlab. Despues de una serie de pruebas, el rango de tiempo es de entre 0.0475 y 0.0460 segundos.

- Octave. Se realizaron la misma cantidad de pruebas que en Matlab y los rangos de tiempo de respuesta se encuentran enrte 0.0775 y 0.0760 segundos.

Como podemos obseerbar el retraso que se genera debido al diseño de nuestro sistema es minimo, lo cual no dio una perspectiva distinta acerca del enfoque con el que abordamos nuestro proyecto, pero esto lo discutiremos más adelante en la sección de conclusiones.

Hasta este punto nuestro sistema se encontraba en una face intermedia, es desir no podiamos capturar y reproducir en tiempo real, sin embargo, el sistema lo pudimos probar en Matlab y pudimos aplicar los efectos a archivos de audio pregrabados y el resultado, cmo es de suponer es exitoso, debido a que al leer de un archivo estatico, podemos obtener información importante, como por ejemplo la duración del archivo. Sin embargo el objetivo no era crear un editor de audio, ni crear un sistema de procesamiento de efectos con alta latencia. Por lo que seguimos relizando pruebas tratando de redicir al maximo la latencia del software, que hasta este putno estaba casi terminado. Y fue en esste punto que el proyecto tomo una vertiente distinta; desidimos realizar las pruebas sobre Small Basic.

Desidimos realizar las pruebas sobre Small Basico, debido a dos factores principales:

- Es un lenguaje facil de implementar y puede ser ejecutado sobre linux sin ningun problema, lo que como ya habiamos mencionado en las secciones anteriores nos facilita la manipulación de la tarjeta de audio.

- Al igual que Matlab y Octave es un lenguaje interpretado, lo que nos mantiene sobre la misma linea en cuanto al tipo de lenguaje; lo que nos permite realizar un punto de comparacion, lo cual discutiremos más adelante.

En estas pruebas con Small Basic, no se diseño una interfaz grafica,

debido a las limitaciones del mismo lenguaje, ya que si bien es sencillo de implementar sobre linea de comandos, crear la interfaz si es un tarea complicada y por razones de tiempo, desidimos, crear nuestros scrpts para la captura, procesamiento y reproducción de audio.

Los resultados de estas pruebas resultaron ser mucho más exitosas de lo que pudimos imaginarnos, ya que el retraso es prácticamente imperceptible tanto para el que utiliza el softwar debido a que no se nota la diferencia de tiempo en que se ejecuta alguna nota musical y en el que se percibe el sonido generado. En este caso no pudimos medir exactamente el tiempo que tarda en realizar el procesamiento completo, pero como ya mencionams, las pruebas respaldan esta afirmación. Cabe mencionar, que las pruebas en los tres lengiajes de programación ejecutados tanto en Linux, como en windows respectivamente, fueron realiadas utilizando los mismos algoritmos, la unica diferencia que existió en las pruebas, fue la forma en que se capturaron y reprodujeron los datos.

En cuanto a los resultados en la interfaz, se lograron de manera parcial, debido que la interfaz se construyo pensando en utilizar Matlab como plataforma para ejecutar las puebas, esta interfaz cumple con lo establecido en nuestros objetivos ya que pudimos implementar la idea de utilizar la biblioteca de funciones que proporciona Touchlin y Web Cam WhiteBoard, las cuales nos permite manipular la interfaz que se encuentra en la PC en una superficie distinta ya sea por medi de un lapiz optico o sobre un panel que simula el pedal. Sin embargo debido a que la implementación final de la captura, procesamiento y reproducción de las señales no se implemento sobre Matlab, no pudimos conjuntar estas secciones del proyecto en uno solo.

4. Conclusiones

La principal conclusio a la que llegamos, despues de haber realizado las pruebas en las que medimos el tiempo de procesamiento de los efectos, es que el problema no es de hardware, sino de software, es decir, el problema radica en las limitaciones que nos impone el propio sistema operativo, ya que es este quien maneja y determina el tamaño del buffer con el que trabaja la tarjeta de audio.

En general, este proyecto nos aportó una visión más amplia de los factores que influyen en los sistemas de tiempo real, como lo es el tiempo de procesamiento y la latencia que este genera; a demás de que caímos en cuenta de que programar aplicaciones en tiempo real para ser implementados sobre ambientes de ejecucion no adecuados resulta ser ineficiente ya que el nivel de latencia generado no pueden eliminarse a niveles aceptables.

Por lo que si deseamos que nuestrosoftware se ejecute en tiempo real sobre el sistema operativo que nuestra PC posee por defecto, deberiamos proporcionar un controlador modificado para que deshabilite el buffer con el que trabaja la tarjeta de audio, o modificar los modulos que de audio en el caso de Linux, lo cual no entra en los objetivos de nuestro proyecto. Ahora bien, creemos que es my probable que ejecutado nuestro software, en un Sistema Operativo monotarea como FreeDOS nuestro software funcionara sin nungun inconveniente, ya que al ser monotarea, nuestro software seria el unico proceso en ejecucion lo que nos garantiza toda la atencion del procesador, además de que FreeDOS no le asigna un buffer a la tarjeta de audio.

5. Trabajo futuro

Migrar nuestro sistema tanto la parte de interfa como de procesamiento a un lenguaje compilado, ya que la velocidad de procesamiento de las instrucciones es mucho mayor y esto nos ayuda a reducir aun más la latencia generada al ejecutar las instrucciones de nuestro sistema. Además realizar las pruebas sobre un Sistema Operativo monotarea, como FreeDOS.

Otra de las adecuaciones que deberemos realizar a nuestro sistema, sin duda es aumentar el número de efectos que debe poseer.

La contruccion de una interfaz interactiva con la herramienta webcamwhiteboar, la cual nos proporcionara la sensacion de utiliar un pedal de efectos.

A demás de las adecuaciones ya mencionadas, este proyecto puede tener varias modalidades de implementación, que si bien son bastante ambiciosas, pueden ser implementadas sin enfrentarse a problemas tan complejos ya que se implementarían con hardware especializado para trabajar el tiempo real y no tienen por qué ser de distribución gratuita y podría verse mas como un buen proyecto de inversión.

Una de las modalidades que pretendemos implementar en este nuevo concepto de pedales de efectos virtuales es integrarlo en la memoria rom de una motherboard y ésta a su vez integrarla al cuerpo de la guitarra, esto facilitara la movilidad del guitarrista tanto en el traslado de los instrumentos ya que no tendrá que preocuparse por dos dispositivos distintos(guitarra y pedal) y en el escenario podrá tener un mayor rango de movimiento ya que en cualquier lugar del escenario en el que se encuentre podrá controlar los efectos de la guitarra. Una desventaja de esta modalidad es que al estar integrado totalmente el pedal de efectos es necesario controlarlo con las manos, lo que es un problema porque si es la guitarra principal se deben hacer los cambios de efectos durante la ejecución de la canción y al tener ocupadas las manos se hace casi imposible realizar el cambio de efecto, sin afectar la secuencia de ejecución. Sin embargo puede funcionar para la segunda guitarra la cual no siempre está sonando y puede realizar los cambios de efecto sin afectar la ejecución de la canción.

Otra de las modalidades que puede presentar este pedal virtual es añadiéndole un hardware que controlara el cambio de efectos y volumen de la señal, manteniendo así el esquema tradicional de un pedal; la ventaja que le podríamos agregar a esta combinación de software y hardware es que el pedal controlara el cambio de efectos de

manera inalámbrica (preferentemente vía bluetooth) de esta manera evitamos estar anclados o restringidos de movimiento por un cable que controle el pedal.

La modalidad que más llama la atención en este proyecto es la de aplicar tecnología wifi al pedal; esto nos abre la puerta a un nuevo concepto de música distribuida, y que al contar con una motherboard, podemos también contar con puertos de red, en especifico wifi, esto nos lleva un paso más allá, ya que si contamos con un ingeniero de audio podrá acceder vía Internet a la interfaz de nuestro pedal de efectos desde cualquier punto geográfico y ecualizar nuestros efectos o cargar en la memoria algún efecto nuevo en tiempo real. O incluso el guitarrista podrá estar en cualquier parte con acceso a Internet y podrá transmitir a cualquier parte su señal de audio.

Referencias

[DFX] niversidad de Las Palmas de Gran Canaria, Graficos 3D por computadora, 2010, http://www2.dis.ulpgc.es/

[SIELO] uan Amenábar, Revista musicla Chilena, Facultad de artes, Uinversidad de Chile, Julio, 2000.

[REN] ergi Jordá Puig, Audio Digital y MIDI, 1997-2003.

[AUDIO] eff Tranter, El como del sonido - Linux, Junio, 1996.

[MAT] belardo Jara, Matlab en Liux, Marzo, 2008.

[DSP] ohn G. Proakis and Dimitris G. Manolakis, Digital Signal Processing, Macmillan Publishing Company, 1988

[FX] ohn Wiley and Sons, Ltd, Digital Audio Effects, Udo Zolzer, 2002.

[GUIM] osé Jaime Esqueda Elizondo, Interfaces Gráficas en Matlab Usando GUIDE, Noviembre, 2002, Universidad Autónoma de Baja California, Unidad Tijuana"

[ADA] rian D. Storey, Using the MATLAB Data Acquisition, Marzo, 2008.

[ADA] iego Orlando Barragán Guerrero, MANUAL DE INTERFAZ GRÁFICA DE USUARIO EN MATLAB, Mayo, 2008.

[MSA] klahoma State, Real-Time Audio Visualization, Noviembre, 2010.

[AEM] epartment of Electrical Computer Engineering, Audio Effects in MATLAB, McGill University, 2000.