kontakt script lenguaje

Gustavo David Ferreyra brakdag@gmail.com

6 de agosto de 2010

Índice general

Capítulo 1

Kontakt Script Lenguaje

1.1. Introducción, conceptos básicos

1.1.1. ¿Qué es KSP?

Es el procesador de código de kontakt. (kontakt Script Processor)

Es uno de las características que tiene kontakt, puede usarse para crear efectos o como una herramienta de composición. Se pueden crear intrumentos con algoritmos inteligentes de procesamiento.

1.1.2. ¿Es un lenguaje de programación? pero Yo soy un músico no un fanatico de la computación!

Si es un lenguaje pero es facil! *Lenguaje de programación* o *scripting*, en estos capitulos explicaré de forma detallada paso por paso para que los novatos en programación de computadora puedan empezar. No es necesario conocer acerca de ningun lenguaje de programación para comenzar a usar KSP, además podés hacer tus propios scripts, o reformar alguno del paquete de scripts que trae kontakt.
Si estás leyendo esto y sos un geek o hacker cracker o lo que sea, y tenés experiencia en programar en varios lenguajes y entornos. Te recomiendo que leas el manual de scripting directamente de kontakt, y no perdás tiempo con esto.
¿Pero qué hace falta para empezar a aprender? ¿Algun software especial?
Necesitas solamente cualquier versión de kontakt el que tengas,configurado con un correcto funcionamiento midi/audio. Y no necesitas nada mas!.

1.1.3. Comenzando - El KSP

Para comenzar con esto, es mejor clarificar algunos conceptos y quizá no estén muy claros.

¿Dónde está el KSP?

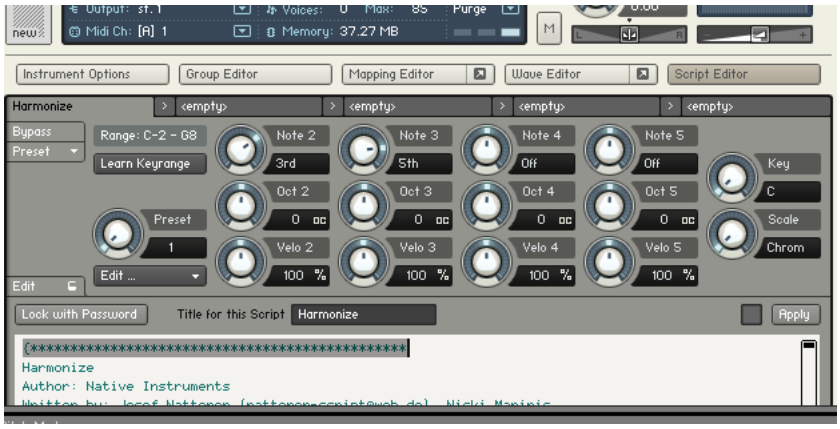


Figura 1.1: El editor de código.

El procesador de código de kontakt es parte de kontakt, y constrituye un elemento importante del flujo del soft. Cuando enviás una señal midi esta recorre varios caminos antes de podes escuchar la señal de audio.
Tocas la nota - se dispara el sample - Grupo de FX - amplificador
- Grupo de Fx de instrumento - Salida del sonido.
el KSP estaría ubicado después de tocar la nota.
Tocas la nota-KSP se dispara el sample - Grupo de FX - amplificador - Grupo de Fx de instrumento - Salida del sonido.

Sin embargo el KSP no es un simple procesador MIDI, ¿Porqué? Porque éste tiene acceso a diversos parametros internos de kontakt, que otros procesadores midi no pueden obterner, por ejemplo el manejo de grupos en un instrumento, samples,efectos internos etc. ¿Bueno, y cuando empezamos con el código? jajaja, buena pregunta, ahora empezamos, primero hay que encontrar donde está el editor de codigo.

1. Primero cargá un instrumento en kontakt.

2. Después hacé click en a la izquierda en un simbolo de llave inglesa (simbolizando un herramienta)

3. Despues ves que tenes varios lenguetas, hacé click en la que dice Script Editor

4. hacé click en Preset, y aparecerá un menú desplegable.

5. hacé click en Harmonization/Hamonize (Aparecerán unos controles en el panel del modulo de scripts.)

6. Después hace click en edit y ohhh sorpresa!!.

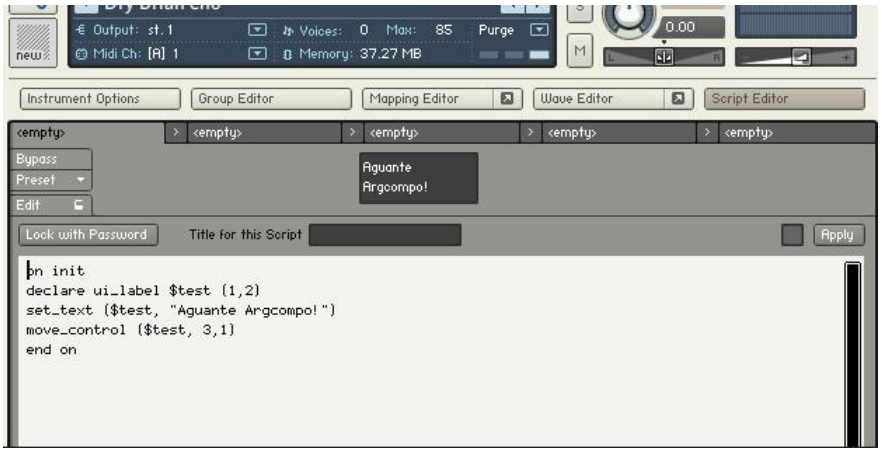


Figura 1.2: Agregando el código

Con eso acabamos de cargar un script y podemos editarlo, examiná un poco el codigo a ver si entendes algo. No hay problema si no entiendes nada, en el transcurso de estos capitulos terminaremos todos aprendiendo a hacer codigo y a modificarlo a nuestro gusto.

1.1.4. Creando nuestro primer codigo KSP en kontakt.

Bueno, tenemos cargado el codigo anterior lo que vamos a hacer es borrarlo, para hacer esto vamos al icono que dice **Script** y selecciones - **Empty** - en el menú desplegable. Esto borra todo el código que habia. Copiamos el siguiente código y lo pegamos. hacemos copy&paste del código que se muestra en la Figura 2.

Notaremos que el led que está cerca de **Apply** se puso de color naranja, indicando que hay algo de script para cargar. Hacé click en **Apply** y verás lo siguiente.

Felicitaciones!!! ya te transformaste en un programador hacker!!

Pero ojo!! todavía no terminaste todo!! donde dice Title for this script podés colocarle el titulo que quieras, descriptivo del script que has hecho y le das enter.

Después obviamente tenemos que grabar el script, y se hace haciendo click en Preset, y después en Save Preset. Escribís un nombre para el archivo (si, ya te diste cuenta!! la extension es .nkp) y lo guardás.

Bueno, creo que esto es lo más básico que hay sobre script, pero seguiré haciendo otros capitulos explicando un poco más a fondo esto del script de kontakt para que lleguemos a hacer tremendos codigos.

1.2. Creando un armonizador

1.2.1. Haciendo sonar notas midi como un acompañamiento simple.

Bueno, ahora ya podemos verdaderamente meternos de lleno en la programación de las cosas básicas de KSP. Vamos a empezar a usar caracteristicas importantes del script como la habilidad de generar notas artificiales *MIDI*, bueno.. empecemos! Abrite el kontakt y cargate un instrumento.! abrí el editor de código y copiá el siguiente código.

```
on note
play_note ( 60,120,0,-1 )
end on
```

Presioná el botón *apply* para que el código sea revisado (si está todo ok entrará en funcionamiento) Tocá un par de notas y escuchá, si seguramente ya te diste cuenta, cada nota que toqués está acompañada por la nota C3 a una velocidad de 120.(la velocidad es la sensibilidad 127 para cuando se toca la tecla con la maxima fuerza y 0 cuando la tecla se presiona muy lentamente)

¿Pero ésto como trabaja?

Siempre que toqués una nota, el KSP procesa una parte específica del código. Éstas partes son llamadas eventos (callbacks). Lo que escribiste antes arriba es llamado evento de nota (note callback). Un evento de nota es una parte del código que es ejecutada siempre que se toque una nota. Bueno ya sabemos que con on note abrimos una sección de codigo para cuando se presione cualquier nota.

play_note(60,120,0,-1) : este es el primer comando que el KSP ejecuta lo vamos a llamar a este comando función. Esta funcíón genera notas midi. En este caso genera la nota *C3* correspondiente al número 60 y a una velocidad de 120. Como se puede obserbar en la tabla correspondiente de notas a numero Midi.

Octave	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-2	0	1	2	3	4	5	6	7	8	9	10	11
-1	12	13	14	15	16	17	18	19	20	21	22	23
0	24	25	26	27	28	29	30	31	32	33	34	35
1	36	37	38	39	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55	56	57	58	59
3	60	61	62	63	64	65	66	67	68	69	70	71
4	72	73	74	75	76	77	78	79	80	81	82	83
5	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107
7	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127				

y con *end on* cerramos dicha sección de código para cuando se presiona una tecla. Veamos un poco mas detalladamente la función play_note.

play_note(número de nota,velocidad,comienzo del sample,duración)

- *número de nota:* Número de nota (0 -127)
- *velocidad:* velocidad con la que va a ser tocada (1 -127)
- *comienzo del sample:* este parametro especifica si hay un desplazamiento del comienzo del sample ponemos 0 para que se toque el sample desde el comienzo.(se especifica en milisegundos)
- *duration:* especifica el largo que la nota va a ser tocada. este parametro acepta estos dos valores especiales:
  -1: deja de tocar el sample cuando se deja de tocar la nota que originó el evento
  0: el sample entero es reproducido.

1.2.2. Creando un simple Octaver

Con el código anterior te habrás dado cuenta que si presionas cualquier tecla con diferentes velocidades siempre iba a sonar C3 a 120, bueno ahora vamos a ir haciendolo un poco más complejo vamos a ir usando más datos midi, para ir procesandolo de forma que sea mas acorde a lo tocado.

bueno, borrá todo lo anterior que tenías (en todo caso guardalo, en el capítulo 1 se explica como hacer eso) dejá todo limpio y copiate este código en el editor de código del **KSP.**

```
on note
play_note( $ EVENT_NOTE - 12,$EVENT_VELOCITY,0,-1 )
end on
```

dando un simple vistaso, y ya estamos entendiendo éste código, porque ya sabemos que esto tiene on note que significa que ese código se va a ejecutar cuando toquemos una nota, también tenemos *play_note* que es una función que ya conocemos.

¿Pero qué hay de nuevo acá?

Obviamente que el bicho extraño que vemos acá es $EVENT_NOTE y $EVENT_VELOCITY, que simplemente cada una es una variable donde se almacena el número de nota tocada ($EVENT_NOTE), y la velocidad con que es tocada (EVENT_VELOCITY), sabemos también que los números de notas estan relacionados con los semitonos, o sea 12 semitonos serian una octava, do do# re re# mi fa fa# sol sol# la la# si (12) si restamos 12 a la nota tocada tocaremos la misma nota a la vez con la misma velocidad pero una octaba abajo, toquemos un poco y escuchemos. Si! ya estamos haciendo magia con script ya tenemos un octaver!
Además en este ejemplo se pudo ver que podemos sumar y restar números usando los signos + y - y que son interpretados correctamente por el KSP
Bueno, pegale una experimentada poné otros número cambia parámetros, es hora de tomarse un relax, y jugar un poco con eso!

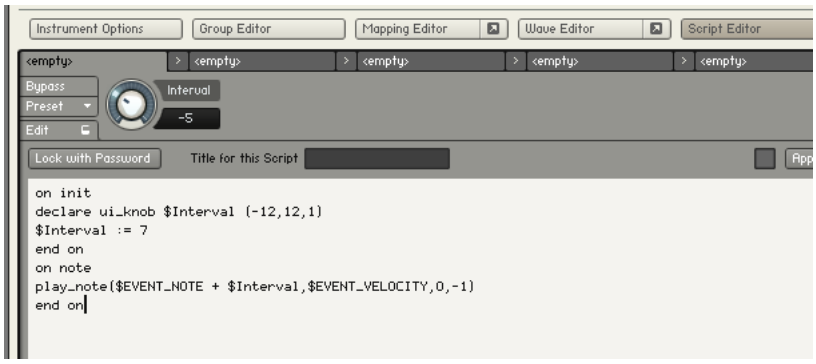


Figura 1.3: código de armonizador en ksp

1.2.3. Creando un pequeño armonizador

Bueno ya creamos un codigo antes, pero sería muy bueno usar un control potenciometro (knob) para especificar cuanta distancia haya entre la nota original y la generada por nuestro código.
veamos un ejemplo, como siempre, borrá todo y copiá el siguiente código.

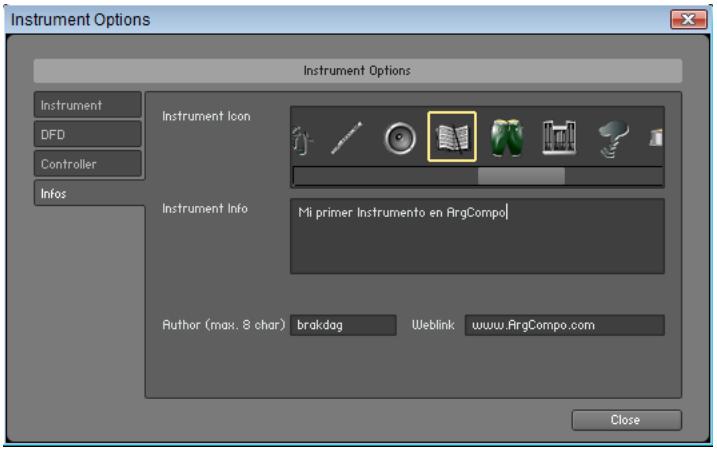
```
on init
declare ui_knob $Interval (-12,12,1)
$Interval := 7
end on
on note
play_note($EVENT_NOTE + $Interval,$EVENT_VELOCITY,0,-1)
end on
```

Le damos click en *apply* y vemos ohh, sorpresa!! apareció un control! como se muestra en la Figura 4.

Bueno en este código tenemos un par de cosas nuevas por ejemplo on init que es un evento que se ejecuta todo lo que haya dentro cuando se comienza a usar el script. Dentro de este evento tenemos la declaración de un control de interfaz gráfica ui declare ui_knob $Interval (-12,12,1), con esto creamos el knob, y la variable que almacena el valor del knob es $interval, que después en el mismo código le dá un valor de 7 ($Interval := 7) si, ya te diste cuenta! el signo := sirve para cargar valores a las variables. El resto del código ya es conocido para nosotros, vemos un poco más como hicimos para crear esta perilla rotatoria.
declare ui_knob $ nombre de variable¿(¡min¿,¡max¿,¡distancia de paso¿)
bueno, el $¡nombre de variable¿es un nombre que le damos para identificarlo y despues poder usarlo en cualquier otra función (siempre precedido por el signo $) ¡min¿: es el mínimo valor que está disponible en el knob, igual que ¡max¿el maximo valor, y simplemente ¡distancia de paso¿es lo que nos indica de cuanto en cuanto va a ir saltando el control, probá un poco juga con esto reemplazá valores inventate algo, es divertido! juga un poco y seguiremos aprendiendo un poco más de todo esto, pero en el próximo capítulo. xD

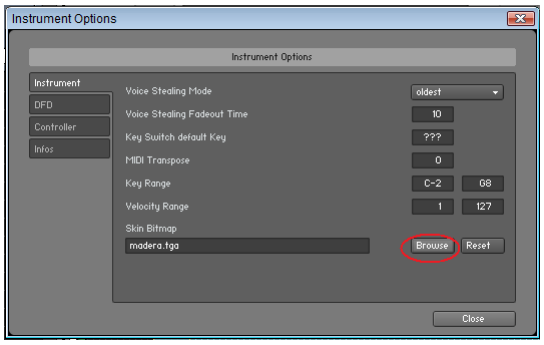
1.3. Personalizando nuestro instrumento

Como muchos estámos interesados en crear nuestro propio instrumento o personalizarlo, voy a salirme un poquito fuera del orden del manual, asi damos un vistaso para editar efectos/apariencia del instrumento. (Esto es solo un adelanto, en capítulos siguientes iremos paso por paso explicando cada detalle.)
Abrí cualquier instrumento o directamente desde el menú donde se ve un diskette, crea un instrumento nuevo. (Cargale unos samples en el sector de MapEditor si ya sabes del tema. En este caso usé un piano de EW Bosendorfer 290)
En el instrumento hacé click en la llave inglesa, luego click sobre *instruments Options*. Hacemos click en la lengueta *infos*, elegimos un icono a gusto y completamos los datos.

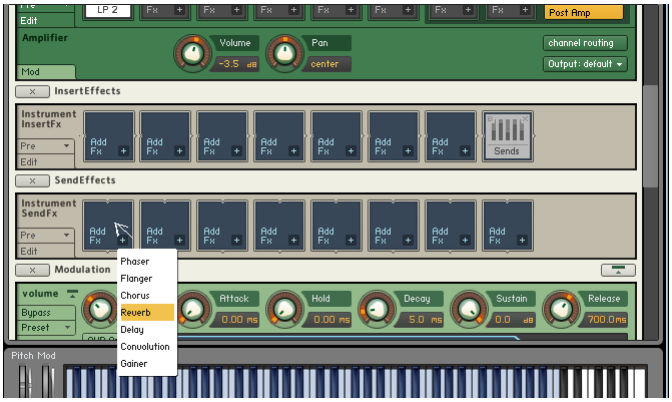


Ahora nos toca cargar la imagen de fondo para el instrumento. El tipo de archivo de imágenes soporta kontakt es .TGA dimensiones 633 x2 25. Ya tengo una imagen editada (La mágia de los tutoriales, parece utilísima esto??

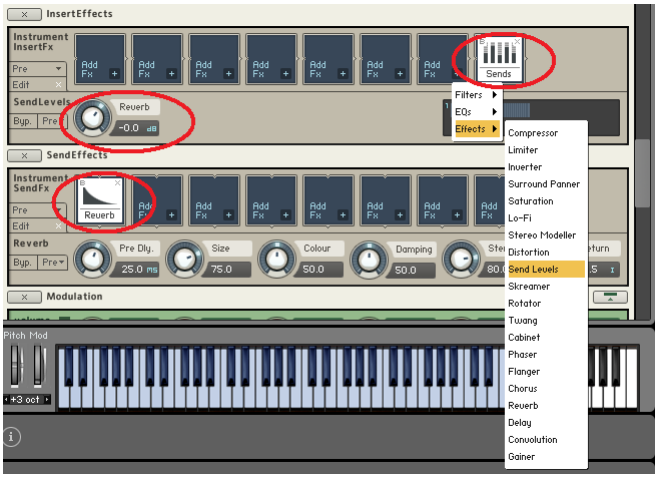
Vamos a la lengueta *Instrument* y hacemos click en *Browse*, para seleccionar nuestra imagen, como se muestra en la imagen siguiente.



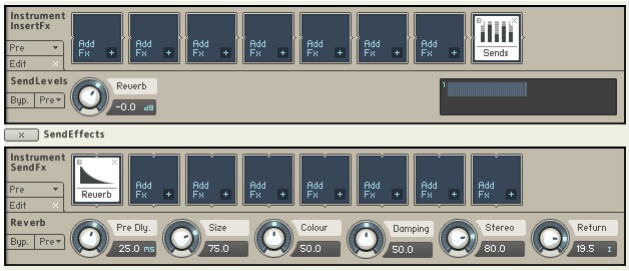
Cargamos ahora un efecto de reverb, para controlar en nuestro instrumento. (puede ser cualquiera)



Luego en el módulo de Instrument *InsertFx* agregaremos un *Send Levels*, para conectar el efecto reverb con la salida de sonido, y le daremos un nivel de 0db.



Entonces en la sección de efectos nos tiene que quedar configurado algo así como se ve en esta imagen.



Perfecto, ya tenemos la imagen cargada y información efectos , ahora vamos nuevamente al editor de código. Agregamos el ésto.

```
on init
make_perfview
declare ui_knob $Reverb (0,100,1)
move_control ($Reverb,1,4)
end on

on ui_control ($Reverb)
_set_engine_par($ENGINE_PAR_SEND_EFFECT_OUTPUT_GAIN,$Reverb * 10000,-1,0,0)
end on
```

Cargamos el código, a continuación se explica un poco que significa cada parte del mismo.

- *on init*:como vimos anteriormente esto es un evento, que hace que se ejecute todo lo que está dentro hasta llegar a end on cada vez que se inicia el código (script/instrumento)

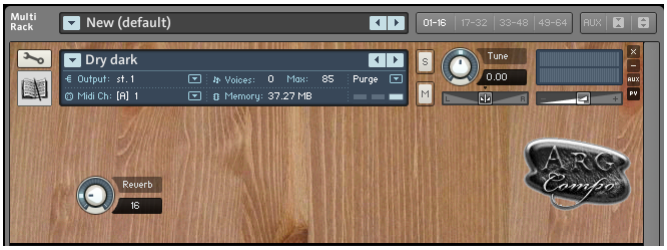
- *make_perfview*: siempre que coloquemos esto, nos carga todos los botones y knob y controles del script directamente en el instrumento, si!! esta es la funcion mágica que hace visibles los controles debajo del instrumento en kontakt.

- declare_ui_knob $Reverb (0,100,1) como viemos en el capitulo anterior esto crea el control knob que es un potenciometro rotatorio 0 es el valor mínimo asignado, 100 es el valor máximo, y 1 es el movimiento mínimo permitido o sea de cuanto en cuanto va a saltar por un movimiento minimo del control.

- move_control ($Reverb,1,4) Simplemente esta funcion sirve para mover los controles en la pantalla, aquí se indica el nombre de la variable del control ($Reverb), la posición en el eje horizontal (1) y la posición en el eje vertical (4) , aumenta hacia abajo, el eje vertical, y el eje horizontal aumenta hacia la derecha.

- on ui_control ($Reverb) este evento ocurre cada vez que movamos cualquier control, en este caso se especifica entre paréntesis, o sea cada vez que se mueva el knob de reverb, se va a ejecutar el codigo que hay adentro, o sea hasta llegar a end on
- _set_engine_par($ENGINE_PAR_SEND_EFFECT_OUTPUT_GAIN,$Reverb * 10000,-1,0,0) Esto hace que se envie un parametro para cambiar el efecto, en este caso el reverb, con los valores provenientes de nuestro knob llamado $Reverb. (Para información detallada leer el manual de script).

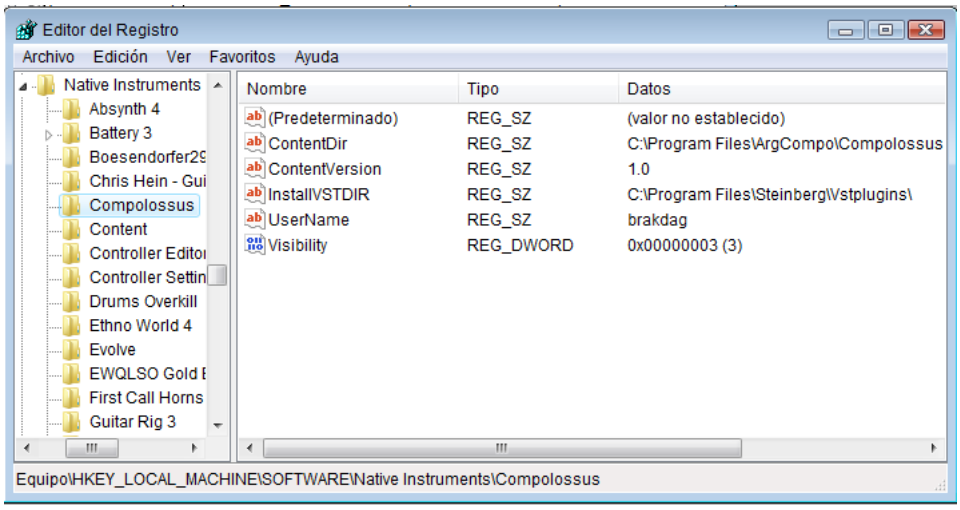
Hacemos click en Apply Cerramos el editor de instrumento y vemos que nos quedará algo así como esto.



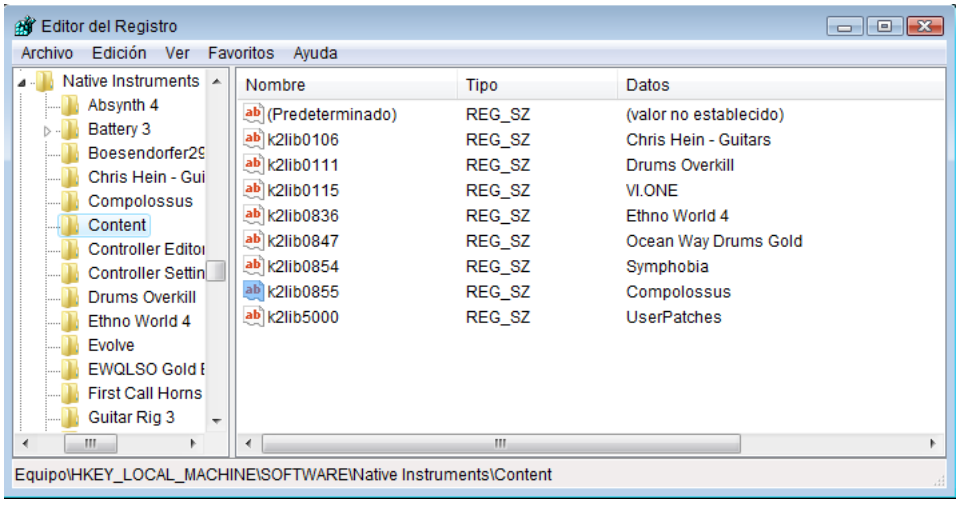
Bueno, espero puedan hacer algo... seguiremos en el próximo capítulo examinando un poco más el código de kontakt. Es fácil !, Ya podremos agregar más efectos!! En unos cuantos capitulos más agregaremos otros controles, que son muy interesantes

1.4. Creando nuestra propia librería en kontakt (Windows)

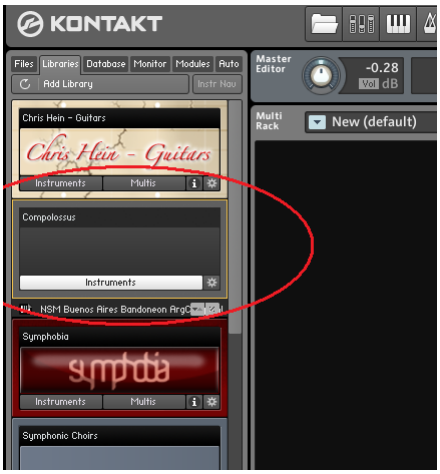
(El siguiente contenido, decidí agregarlo pero no figura en ningun manual, esto es información interna de Native Instruments, y ha sido obtenida a través de procesos mios ingeniería inversa por lo que no puedo garantizar su funcionamiento del 100En principio para crear una librería metemos todos los instrumentos con extensión .nki en una carpeta llamada *Instruments.* Esa carpeta generalmente la tenemos que tener dentro de una ruta donde generalmente se instalan los instrumentos (ej: c:/Program Files/ArgCompo/COmpolossus) Yo agregué para este caso el instrumento de bandoneon, de Argcompo. Muy bien, pero esto de por sí solo no es nada, tenemos que hacer que kontakt lo reconozca como librería.Para que lo reconozca tenemos me agregar un par de claves al registro. Para que esto ocurra ejecutamos la aplicación *Regedit* (Desde el menú ejecutar de windows.). Luego buscamos la carpeta correspondiente a Native Instruments.
`HKEY_LOCAL_MACHINE/SOFTWARE/Native Instruments`
Crearemos una nueva carpeta (Clave) llama Compolossus (Es sólo un ejemplo, puede usarse cualquier nombre que le queramos dar a nuestra librería) Y completamos los siguientes datos como se muestra en la imagen.



Ya que tenemos cargados los datos básicos de la librería, podemos cargarla al contenido de kontakt para que la lea. Para esto agregamos una nueva clave que tiene que tener como nombre formato similar a k2libXXXX donde las XXXX es un valor numerico entre 0 y 5000, y dentro de esta clave de valor alfanumerico, agregamos el nombre de nuestra librería. (se agrega dentro de la carpeta content dentro de Native Instruments)



Bueno, cerramos el regedit, ya que no lo vamos a usar mas, y abrimos kontakt, vamos a la sección de librerías, y ohhh!! está nuestra librería!!!



Además también aparece la lista de instrumentos y efectivamente como habíamos copiado el bandoneon de ArgCompo aparece éste en la lista de instrumentos. Crear imágenes es bastante sencillo. Se busca uno una imagen en el google, la encaja con el paint o cualquier soft similar en un rectángulo de 214*100 pixels(si lleva texto, que éste esté entre (8,28) y (208,78) para que no lo tape la ventana del nombre) y se guarda con el nombre Wallpaper.jpg dentro de la carpeta principal de la librería. Y ya está.

1.4.1. en MAC

1. ir a esta dirección Macintosh HD/Library/Preferences
2. localizar un archivo .plist por ejemplo com.native-instruments.Symphobia.plist y copiar y pegarlo a este.
3. cambiar el nombre de este a el nombre de su libreria ej: com.native-instruments.Tonehammer Waterphone.plist
4. abrir este con Plist Pro y editarlo borrando todas las entradas excepto : a. ContentDir y b. Visibility (el cual necesita ser puesto en 3)
5. ahora abrir com.native-instruments.Content.plist
6. crear un nuevo sibling y nombrarlo así k2libxxxx (donde xxxx usa solo 4 digitos numerales, no colocar el mismo numer de uno de los ya existentes.. Por ejemplo k2lib0001 y el nombre es Tonehammer Waterphone
7. Coloca library's localizacion bajo ContentDir en el com.native-instruments.Tonehammer Waterphone.plist
8. Abrir Kontakt

1.5. Tocar en vivo con kontakt bajo windows

Generalmente si nuestro teclado o sintetizador no cumple con los requerimientos en calidad de sonido para tocar en vivo buscamos una alternativa en módulo de sonido, y que alternativa más economica que usar una pc, y en este caso con un sampler el famoso kontakt.

En el proceso de configuración tenemos que tener en cuenta algunos detalles, para no pasar estragos mientras uno toca en vivo, en este caso es una previa configuración de todos los sonidos y precarga. Y la configuración correcta de la PC, para que no se interrumpa el sistema.

- Gestión de energía: Todos sabemos que windows siempre estuvo bastante enfocado al tema de gestión de energía y por defecto el sistema se suspende determinado tiempo si no se produce actividad en los dispositivos de entrada. Para windows los unicos dispositivos de entrada que reconoce como actividad de usuario son el teclado y el mouse, por lo mismo por más que estemos tocando midi, no reconocerá ningun movimiento y accionará la suspensión del sistema o el apagado lo que nos producirá el corte del sonido, y una enorme perdida de tiempo en la recarga de samples en el peor de los casos, tocando en vivo esto es un completo desastre que ocurra.
- Gestión de mantenimiento: Lamentablemente, windows si o si detecta que la computadora al no tener movimiento de mouse o teclado, está en un estado idle o de bajo uso, por lo que comienza algunas tareas propias

del mantenimiento, en caso de windows 7 o vista comienza tareas de indexación de archivos para mejorar las búsquedas y otro tipo de tareas, que nos consumen ampliamente el ancho de banda del disco duro, lo que nos afecta directamente al sistema DFD (direct from disk) de kontakt y irremediablemente nos produce cortes perdida de reproducción de samples etc, un tremendo problema a la hora de tocar en vivo. evidentemente si esto no está desctivado puede producir problemas de cortes clipeos, etc, se aconseja para remediar esto, subir el uso de memoria caché de DFD en kontakt, al maximo para hacerlo menos dependiente del DFD. Aunque con sonidos de calidad de 24 bits, es como casi nada. ya que son muy pocos milisegundos por sample.

- Desinfección: Los virus ultimamente troyanos etc, tienen un funcionamiento un poco diferente a lo que ocurria en el pasado, antes los virus tenían un gran uso de cpu los que nos ponía la maquina lenta y lo detectabamos y moría, actualmente los virus son un poco más inteligentes y esperan ese estado de idle, o digamos que el usuario de la pc no está activo para comenzar a realizar sus funciones que pueden ser de diferentes tipos, además las actualizaciones automaticas de empresas como java, adobe, flash, etc, nos instalan este tipo de aplicaciones que hacen generalmente eso, reconstruyen información sin afectarnos o sea que trabajan cuando no estamos, el problema es que si estamos! y queremos tocar en vivo. Y esto nos afecta.

- Gestión de memoria : Cada vez que cargamos un sample, mediante el sistema DFD va a un buffer o a un espacio de memoria en la memoria RAM, por lo que si no disponemos de RAM, actua un sistema SWAP, donde windows emula la ram faltante con un archivo que crea en el mismo disco duro. Por lo que el proceso de reproducción se transforma en algo asi.

1. * Lee archivo con sistema DFD desde disco duro
2. * Reescribe datos de la memoria ram no usados para hacer espacio
3. * Carga el sample en la RAM, reproduce el sonido.

Todos sabemos que la velocidad de escritura de datos en discos duros no es los suficientemente rápida por lo que, nos encontraremos si estamos muy cortos de memoria ram con cortes que se deben a la incapacidad de realizar estas tareas, por lo que si queremos que nuestro sistema esté estable, necesitamos evitar que se use toda la RAM y windows requira realizar este tipo de acciones.

Una vez superados todos estos inconvenientes, podemos decir que tenemos un sistema estable para tocar en vivo, pero

¿A que le llamamos un sistema estable?

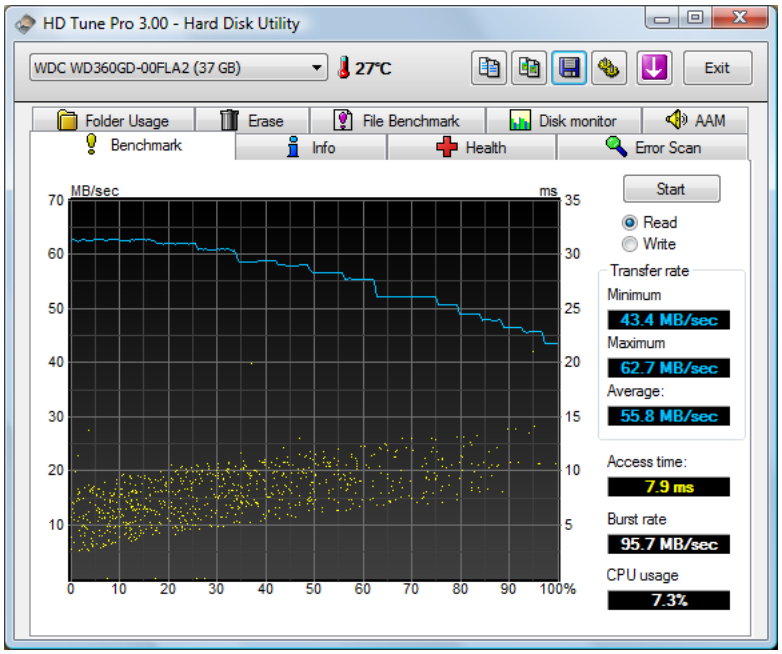
- Ancho de banda fijo de acceso a disco duro (windows no lo garantiza)

- Porcentaje de procesador diponible fijo, lo que proviene de una limpieza de virus, y de software automatico.

- Cantidad de memoria ram disponible fija. Si se cargan softwares que dan gran uso a la memoria ram ocurrirá lo antes explicado.

Bueno, perfecto, si se cumple con cada uno de estos requisitos tendremos un sistema optimo para tocar en vivo, y les aseguro que nunca se les va a colgar nada, vale la pena investigar y testear este tipo de cosas para tener una muy buena calidad de sonido. En otro momento daré unos consejos para solucionar cada uno de estos problemas, hasta pronto.

1.6. Máximo número de samples en kontakt sin producir cortes.



Todos sabemos que es kontakt, es un sampler de native instruments, que bien puede usarse en aplicación standalone, o como plugin.

Los límites de este software estan relacionados con la velocidad de acceso a nuestro disco duro y la cantidad de memoria RAM disponible.

Para testear la cantidad de ram que tenemos disponible para aplicaciones, podemos fijarnos desde las propiedades de nuestro Equipo. Y para testear la velocidad de transferencia que tiene nuestro disco duro necesitamos descargar una aplicación llamada HD tune PRO (http://www.hdtune.com/), que lo que hace es medirnos nuestra velocidad de lectura de disco duro en mb/s. En mi caso poseo una unidad de disco externa (USB 2.0) y la velocidad que me marca de lectura es de 30mb/s.

Generalmente los WAV que carga kontakt son de 16 bits o 24 bits, con una frecuencia casi igual de 44.1 khz. Lo que ocurre es que dependiendo de la calidad

de los samples dependera la cantidad máxima que se puede reproducir. Esto se debe a que kontakt usa el sistema DFD (direct from the disc)

1.6.1. ¿Que es DFD?

DFD (Direct from the disc) es un sistema que carga, solamente la primera parte de las muestras wav (milisegundos) en la memoria ram, y el resto lo lee desde el disco duro.

para la reproduccion de 1 sample con diferentes valores.

24 bits	44.1khz stereo	2116kbps/sample
16 bits	44.1khz mono	705kbps/sample
16 bits	44.1khz stereo	1410kbps/sample

30mb/s *1024kb/mb = 30720kbps

24 bits /44.1khz stereo	30720kbps / (2116kbps/sample)	14 samples.
16 bits /44.1khz mono	30720kbps / 705kbps/sample	43 samples.
16 bits /44.1khz stereo	30720kbps /1410kbps/sample	21 samples.

PD: a prestar un poco más de atención a los bits.