Síntesis Sustractiva con Control de Ratón en SuperCollider.

Autor: Carlos Arturo Guerra Parra
Código fuente del patch de SuperCollider

Introducción:

> Este trabajo presenta tres enfoques distintos para implementar una síntesis sustractiva en SuperCollider. El sonido original es generado mediante ruido blanco, y es posteriormente filtrado utilizando un filtro pasa-banda. La frecuencia central y el factor de calidad del filtro son controlados en tiempo real a través de la posición horizontal y vertical del ratón, respectivamente.
> Propuesta 1: Enfoque Clásico

La primera propuesta sigue un enfoque tradicional, utilizando el método `SynthDef` para definir un sintetizador. Aquí, el ruido blanco se filtra y se modula utilizando la posición del ratón. Es una implementación directa, fácilmente comprensible aunque difícilmente modificable en entornos de tiempo real.

```
(
SynthDef(\ruido, {
    var freq = MouseX.kr(100, 3000, \exponential);
    var rq = MouseY.kr(0.01, 0.5, \exponential);
     var sonido = WhiteNoise.ar(1!2);
     sonido = BPF.ar(sonido, freq, rq);
    sonido = sonido * rq.lincurve(0.01,1,10,1,-20);
     Out.ar(0, sonido);
}).add;
)

x = Synth(\ruido)
x.free
```

Analicemos cada línea de código del sintetizador:

1. **Control de Frecuencia y Factor de Calidad:**
    a. `freq = MouseX.kr(100, 3000, \exponential)`:
    Esta línea define la variable `freq` que controla la frecuencia central del filtro pasa-banda. La frecuencia se modula en tiempo real con la posición horizontal del ratón. El rango de frecuencia es de 100Hz a 3000Hz para que esté dentro de lo audible por la mayor parte de las personas y al mismo tiempo no haga daño a los oídos. Se modula de manera exponencial para ofrecer un control más musical y natural.
    b. `rq = MouseY.kr(0.01, 0.5, \exponential)`:
    Aquí, `rq` representa el factor de calidad del filtro pasa-banda. Se modula con la posición vertical del ratón en un rango que va desde 0.01 a 0.5, también de manera exponencial. Se ha cuidado en que el valor inferior no sea

demasiado próximo a 0, ya que, al ser el inverso de Q, el valor de este sería infinito, lo que podría provocar efectos indeseados en el filtro `BPF.ar`, tal como se indica en su documentación.

2. **Generación del Ruido Blanco:**
    a. `sonido = WhiteNoise.ar(1!2)`:
    Se utiliza el generador `WhiteNoise` para crear ruido blanco. El término `!2` indica que se generan dos canales de audio (estéreo) con dos versiones de `WhiteNoise.ar`. Esto es lo que en SuperCollider se denomina "Expansión Multicanal", permitiendo que el ruido se escuche, en este caso, tanto en el canal izquierdo como en el derecho con distinto contenido y evitando así el efecto monofónico.
3. **Filtrado del Ruido Blanco:**
    a. `sonido = BPF.ar(sonido, freq, rq)`:
    La señal de ruido blanco (`sonido`) se pasa a través de un filtro pasa-banda (`BPF`) con la frecuencia central y el factor de calidad modulados por la posición del ratón, como se definió anteriormente.
4. **Modulación de la Amplitud:**
    a. `sonido = sonido * rq.lincurve(0.01,1,10,1,-20)`:
    La señal filtrada se multiplica por un factor modulado por el valor `rq`. Esta línea asegura que, a medida que el el inverso del factor de calidad (`rq`) aumenta, la amplitud de la señal se modula de manera adecuada para que el sonido no se vuelva demasiado fuerte según aumenta la banda del filtro. La función `lincurve` es utilizada para lograr una curva adecuada para esta modulación. El valor -20 responde a la forma de la curva y ha sido elegido por ensayo error.
5. **Salida de Audio:**
    a. `Out.ar(0, sonido)`:
    Finalmente, la señal procesada se envía a la salida de audio principal (canal 0) para ser escuchada. Puesto que `sonido` es un array stereo, sus dos componentes se reproducirán en los canales 0 y 1 respectivamente, que corresponden a la salida stereo del sistema operativo.

Propuesta 2: Enfoque con Ndef

El segundo enfoque utiliza `Ndef`, permitiendo una gran flexibilidad para mapear y configurar parámetros en tiempo real. Este enfoque es ideal para situaciones de live coding, ya que facilita la manipulación en tiempo real de los nodos de sonido:

```
(
~ruido = { arg freq = 440, rq = 1, amp = 1;
      var sonido = WhiteNoise.ar(1!2);
      sonido = BPF.ar(sonido, freq, rq);
};
)
Ndef(\ruidoFiltrado, ~ruido)
Ndef(\ratonX, {MouseX.kr(100, 3000, \exponential)})
Ndef(\ratonY, {MouseY.kr(0.01, 0.5, \exponential)})
Ndef(\ruidoFiltrado).map(\freq, Ndef(\ratonX))
```

```
Ndef(\ruidoFiltrado).map(\rq, Ndef(\ratonY))
Ndef(\ruidoFiltrado).play
Ndef(\ruidoFiltrado).stop(6)
```

A nivel de lógica de síntesis, no añade nada al enfoque de la propuesta 1. Puede resultar más prolijo y difícil de entender a primera vista, pero es sin duda un enfoque más adecuado para su uso en vivo, ya que permite muy fácilmente modificar el sintetizador “en caliente”, sin tener que liberar y crear objetos explícitamente.

Propuesta 3: Enfoque Compacto (SCTweets)

La tercera propuesta presenta dos versiones compactas inspiradas en el formato SCTweets, práctica que se estuvo de moda hace unos años en el mundo de SuperCollider. El fin de estos trabajos era el de condensar en el menor número de caracteres un patch útil e interesante con código de SuperCollider y que pudiera ser compartido en un sólo tweet. Aquí va la propuesta para este trabajo.

- **3.a Versión completa:**
  Esta versión utiliza un generador de ruido `GrayNoise` y modula la amplitud del sonido según la posición vertical del ratón. La elección de `GrayNoise` en lugar de `WhiteNoise` se ha hecho para ganar un solo carácter. Existen otras decisiones en la misma línea, como minimizar el uso de variables introduciendo los objetos directamente en los parámetros o prescindir de espacios, aunque todo ello sea a costa de la pérdida de legibilidad y claridad del programa:

  ```
  play{y=MouseY.kr(0.01);BPF.ar(y.lincurve(0.01,1,10,1,-20)*GrayNoise.ar,MouseX.kr(99,3e3),y)}
  ```

- **3.b Versión superreducida:**
  Con sólo 60 caracteres, esta versión se queda con lo esencial del programa, mostrando la capacidad de SuperCollider para realizar tareas complejas de manera extremadamente concisa:

  ```
  play{BPF.ar(GrayNoise.ar,MouseX.kr(99,3e3),MouseY.kr(0.01))}
  ```

Conclusión:

Las tres propuestas solo son una muestra posible de la versatilidad y potencia de SuperCollider como herramienta para la síntesis de sonido. Dependiendo del contexto y de las necesidades del usuario, cada enfoque tiene sus propias ventajas. Se han dejado de lado, por razones de espacio, técnicas como uso de patrones y rutinas que podrían dotar de texturas más interesantes al mismo requerimiento de esta tarea. El enfoque clásico es directo y fácil de entender, el enfoque con `Ndef` es altamente flexible y perfecto para live coding, y el enfoque compacto (en sus dos versiones) muestra la eficiencia y concisión que puede alcanzar SuperCollider.