Música por computadora
Ernesto Romero y Hernani Villaseñor
Centro Multimedia 2012

Sesión 6

Tdef, boleanos, if, switch.

Tdef

Veamos que es y como funciona la herramienta Tdef de SuperCollider. Tdef viene de Task definition, en español definición de tarea. Tdef es útil para realizar precisamente tareas que requieren varios pasos en su desarrollo. Es un tipo de rutina, donde rutina es el término general empleado en el ambito de los lenguajes de programación para designar un paquete de acciones que han de realizarse en cierto orden y en determinado tiempo. Para hacer un Tdef se necesita escribir primero su forma básica:

Ejemplo 1

```
Tdef(\baladi, { "no soy baladi".postln })
Tdef(\baladi).play
```

El Tdef necesita tener un nombre que se escribe antecedido de una diagonal invertida: \nombre. Esto es un símbolo. El Tdef anterior se llama \baladi. Después del nombre hay que agregar una función; habremos de usar las llaves `{}` para esto. Dentro de las llaves escribimos lo que queremos que haga el Tdef. El Tdef del ejemplo 1 tiene por tarea imprimir en la post window el mensaje "no soy baladi" una vez. Para que el Tdef realice su tarea hay que enviarle el mensaje .play. Podemos pedirle al Tdef que realice más de una acción en su tarea. Imprimir por ejemplo 2 mensajes. Observar que hay que poner punto y coma ; al final de cada acción.

Ejemplo 2

```
Tdef(\baladi, {
"no soy baladi".postln;
"o si?".postln;
})
```

Podemos pedirle al Tdef que espere cierta cantidad de tiempo entre la realización de las acciones.

Para esto le enviamos el mensaje .wait a un número que representa la cantidad de tiempo en segundos que queremos que espere.

Ejemplo 3

```
Tdef(\baladi, {
     "no soy baladi".postln;
     2.wait; // espera dos segundos
     "o si?".postln;
     1.wait; // espera un segundo
     "%&*^@#$*&*".postln;

{SinOsc.ar(SinOsc.kr(10,0,100,400))*Line.kr(1,0,2,doneAction:2)}.play
});

Tdef(\baladi).play
```

Ahora bien, si juntamos la estructura n.do{ } con el Tdef podemos repetir n veces las acciones que queramos.

Ejemplo 4

```
Tdef(\baladon, {
     8.do{
{SinOsc.ar(SinOsc.kr(10,0,100,400))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;
     0.25.wait
     };
     16.do{
{SinOsc.ar(SinOsc.kr(15,0,100,700))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;
     0.125.wait
     }
});

Tdef(\baladon).play
```

Podemos anidar funciones y hacer infinitas repeticiones de ellas escribiendo inf.do

Ejemplo 5

```
Tdef(\baladon, {
     inf.do{
     2.do{

{SinOsc.ar(SinOsc.kr(10,0,100,400))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;
     0.25.wait
```

```
     };
     2.do{

{SinOsc.ar(SinOsc.kr(15,0,100,700))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;
     0.25.wait
     };
     4.do{

{SinOsc.ar(SinOsc.kr(10,0,100,400))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;
     0.125.wait
     };
     1.do{

{SinOsc.ar(SinOsc.kr(15,0,100,700))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;
     0.5.wait
     }
     }
});

Tdef(\baladon).play

Tdef(\baladon).stop
```

Booleanos y condicionantes

Los conceptos de boleanos y condicionantes nos ayudarán a introducir estructuras más complejas en nuestros Tdefs.

Los booleanos emplean operadores relacionales (e.g. <, >, ==) entre datos para obtener un valor lógico de verdadero o falso.

Ejemplo 6

```
4 < 8 // ¿Es cuatro menor que ocho? true
4 > 8 // ¿Es cuatro mayor que ocho? false
2 == 3 // ¿Es dos igual a tres? false
2 != 3 // ¿Es dos diferente de tres? true
```

El simbolo doble == pregunta si los valores son iguales. Es diferente del simbolo sencillo = que nos sirve para asignar valores a variables.

```
a = 10 // a es igual a 10
a == 10 // ¿Es a igual a 10?
```

Conociendo esto podemos usar los booleanos como condición para la ejecución de una función en combinación con la noción if.

Para usar if necesitamos un booleano, una función a ejecutar en caso de que el booleano sea verdadero y otra función a ejecutar en caso de que el booleano sea falso.

```
if(true, {"neto"}, {"cabula"})
if(false, {"neto"}, {"cabula"})
if(4 < 8, {"neto"}, {"cabula"})
if(4 > 8, {"neto"}, {"cabula"})
if(2 == 3, {"neto"}, {"cabula"})
if(2! = 3, {"neto"}, {"cabula"})
```

Recordemos ahora la iteración que viene implementada en las funciones:

```
10.do{|i| i.postln}
```

Si evaluamos el argumento *i* con una condicionante podemos hacer cosas como:

```
10.do{|i| if(i<5,{"si es menor".postln}, {"nel, no es
menor".postln})}
```

Y metiendolo dentro de un Tdef:

```
Tdef(\menor, {
10.do{|i| i.postln;
if(i<5,{"si es menor que 5".postln}, {"nel, no es menor que
5".postln});
0.5.wait;
}
})

Tdef(\menor).play
```

Podemos pedir que solo cuando *i* sea igual a cierto valor se ejecute una acción.

```
Tdef(\si, {
inf.do{
4.do{|i| i.postln;
if(i==0,
{{SinOsc.ar(SinOsc.kr(2,0,500,700))*Line.kr(1,0,0.1,doneAction:2)}.pl
ay;},
{{SinOsc.ar(SinOsc.kr(15,0,100,500))*Line.kr(1,0,0.1,doneAction:2)}.p
lay;}
);
0.5.wait;
}
}
})
```

```
Tdef(\si).play
Tdef(\si).stop
```

Modulo %

El concepto de modulo nos limita la cantidad de números que tenemos para contar. Por ejemplo, si decimos que estamos en modulo 5 solo tenemos 5 números para contar: 0, 1, 2, 3 y 4. Como ven se empieza a contar desde el cero. Si tuvieramos que contar hasta el 5 ya no podemos por que solo tenemos hasta el 4. En ese caso tenemos que regresar al cero.

Entonces una enumeración en modulo cinco quedaría así:

```
0%5=0
1%5=1
2%5=2
3%5=3
4%5=4
5%5=0
6%5=1
7%5=2
8%5=3
9%5=4
10%5=0
```

Si usamos el modulo % dentro de un Tdef podemos hacer ciclos como el siguiente:

```
Tdef(\mod, {
     inf.do{|i| (i%8).postln;
     0.5.wait;
     }
})

Tdef(\mod).play
Tdef(\mod).stop
```

Switch

Switch es una especie de condicionante if pero que puede verificar la igualdad entre más de un valor.

```
Tdef(\si, {
     inf.do{|i|
switch((i%8).asInteger, 0,
{{SinOsc.ar(345)*Line.kr(1,0,0.1,doneAction:2)}.play;},
1,{{SinOsc.ar(1345)*Line.kr(1,0,0.1,doneAction:2)}.play;},
2,{{Pulse.ar(35)*Line.kr(1,0,0.1,doneAction:2)}.play;},
3,{{WhiteNoise.ar(0.7)*Line.kr(1,0,0.1,doneAction:2)}.play;},
4,{{SinOsc.ar(345)*Line.kr(1,0,0.1,doneAction:2)}.play;},
5,{{Pulse.ar(75)*Line.kr(1,0,0.1,doneAction:2)}.play;},
6,{{Pulse.ar(20)*Line.kr(1,0,0.1,doneAction:2)}.play;},
```

```
7,{{WhiteNoise.ar(0.7)*Line.kr(1,0,0.1,doneAction:2)}.play;}
);
0.2.wait;
}
})

Tdef(\si).play
Tdef(\si).stop
```