Envelope

¿Qué conceptos nuevos necesitaremos conocer?

¿Qué es un envelope o envolvente?

El **envelope** va a determinar la variación de los niveles de una señal a lo largo del tiempo. Para nosotros, va a ser una herramienta que nos permitirá **dar forma/esculpir el sonido** con el que estemos trabajando, desde su volumen a su frecuencia. En lo que a síntesis de sonido se refiere, el envelope es una curva que nos va a permitir controlar algún parámetro de una señal (Wilczek, 2022). Ya lo hemos utilizado anteriormente, pero de una manera muy sencilla cuando utilizamos el "**line~**" para suavizar los cambios de volumen en la "Practica 2: nuestro primer patch sonoro". En este caso utilizamos una sola rampa. ¿Os acordáis?

https://gifer.com/embed/cih

Figura 1. Tortuga deslizándose por una rampa.

Los envelopes se componen por **varios segmentos** y en cada segmento encontraremos una rampa diferente. Vamos ver el **envelope ADSR**, que es uno de los mas utilizados. Este envelope consiste en cuatro segmentos: **A=attack, D=decay, S=sustain y R=release.**

Copyright 2026

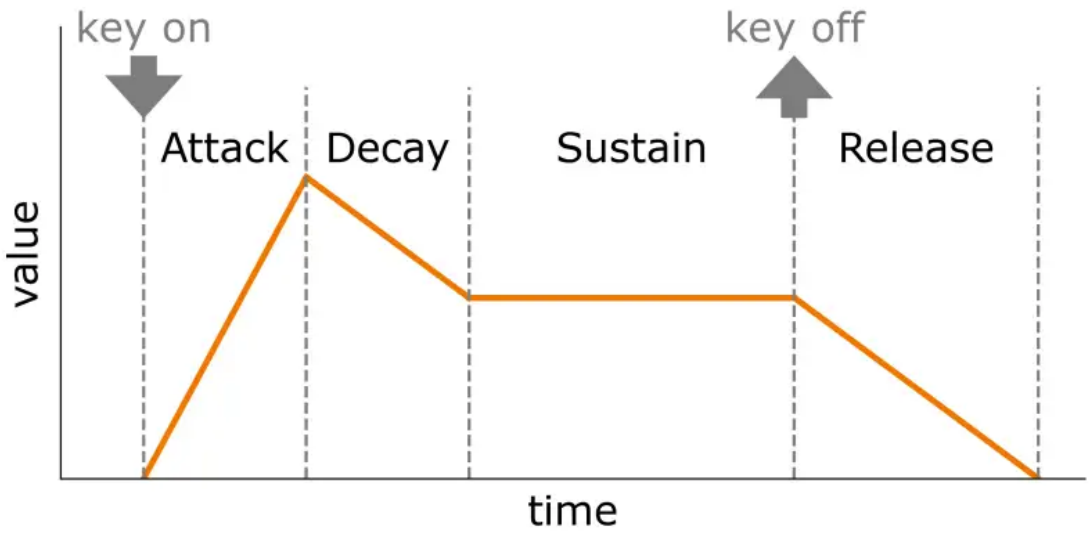


Figura 2. ADSR envelope

El **Attack** es el segmento en el que el valor aumenta desde el mínimo hasta el máximo que va a alcanzar el envelope. El **Decay** es el segmento en el que el envelope cae desde el pico máximo que se alcanza en el Attack, hasta un valor que se mantendrá constante en el segmento conocido como **Sustain**. El **Realease** es el último segmento del envelope en el cual el valor desciende a 0.

Por ejemplo, utilizaremos el ADSR para controlar la amplitud de una señal, en este caso estaremos dando forma al volumen de esta señal. Aqui teneis un ejemplo de ADSR modulando la amplitud:

https://www.youtube.com/embed/k3Pc5_V_HYY?start=21

En el video anterior podéis escuchar cómo cambia el sonido al modificar el tiempo de duración de cada segmento.

Ahora vamos a aprender como hacer envelopes en Pure Data

¿Qué elementos nuevos de Pure data necesitaremos conocer?

Delay

Este objeto nos va a permitir enviar un bang después de un tiempo de espera determinado. Digamos que nos va a permitir retener y **retrasar el envío** de un bang.



El **tiempo** que se retrase el bang lo configuraremos en el **argumento** del objeto o a traves del **inlet derecho**. A traves del inlet izquierdo también podemos configurar el tiempo e iniciar el delay, sin embargo cuando configuremos un tiempo a traves del inlet derecho el delay no empezara hasta recibir un bang en el inlet izquierdo Recordar lo que vimos en la lección "Elementos básicos" sobre la entra caliente de un objeto y las entradas frías. Por defecto la unidades son **milisegundos**, si queremos trabajar con otra unidad como por ejemplo segundos, tendremos que especificarlo en el mensaje que configura el tempo del delay. Si queremos **detener** el delay enviaremos un mensaje de **stop**.

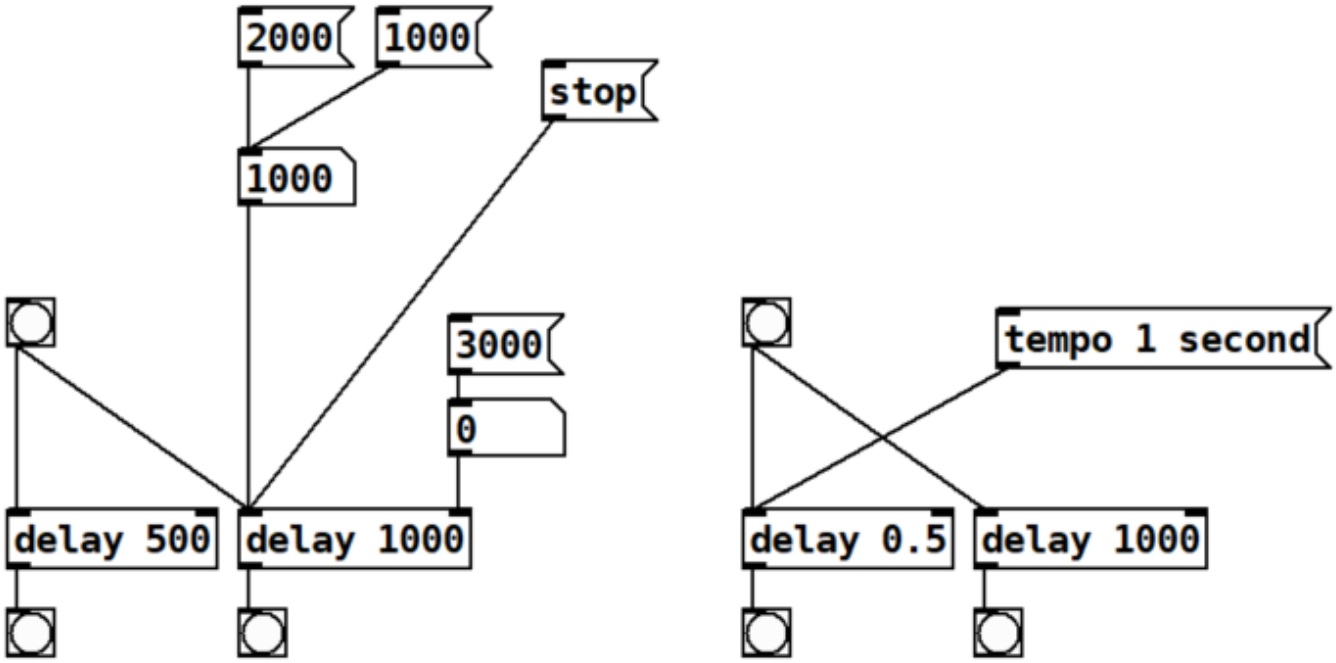


Figura 3. patch *delay-demo.pd*

En la figura 3, el primer delay retrasara el bang medio segundo y el segundo delay 1 segundo. Tras mandar el mensaje configurando el tempo en segundos el tercer delay retrasará el bang medio segundo, si no enviamos ese mensaje configurando el tempo, el delay se retrasará medio milisegundo.

Delay + line

Como vimos anteriormente el objeto "**line~**" nos permite crear **rampas**. Crear una **progresión lineal** desde un valor a otro en un tiempo determinado. ¿Os acordáis? lo vimos en la "Practica 2: nuestro primer patch sonoro".

https://gifer.com/embed/cih

Figura 4. Tortuga deslizándose por una rampa.

Combinando "**delay**" y "**line**" vamos a poder **combinar rampas** para crear progresiones mas complejas, con diferentes direcciones. Por ejemplo ir de 1 a 0.5 en 500 milisegundos y al llegar a 0.5, ir a 0 en 800 milisegundos.

https://giphy.com/embed/l0HluCIeReFkeWAQ8

Figura 5. Combinación de varias rampas.

Vamos a ver cómo crear ahora un **envelope ADSR** con Attack, Decay, Sustain y Release en pure data, podéis abrir el patch *line-ADSR.pd* para probar su funcionamiento:

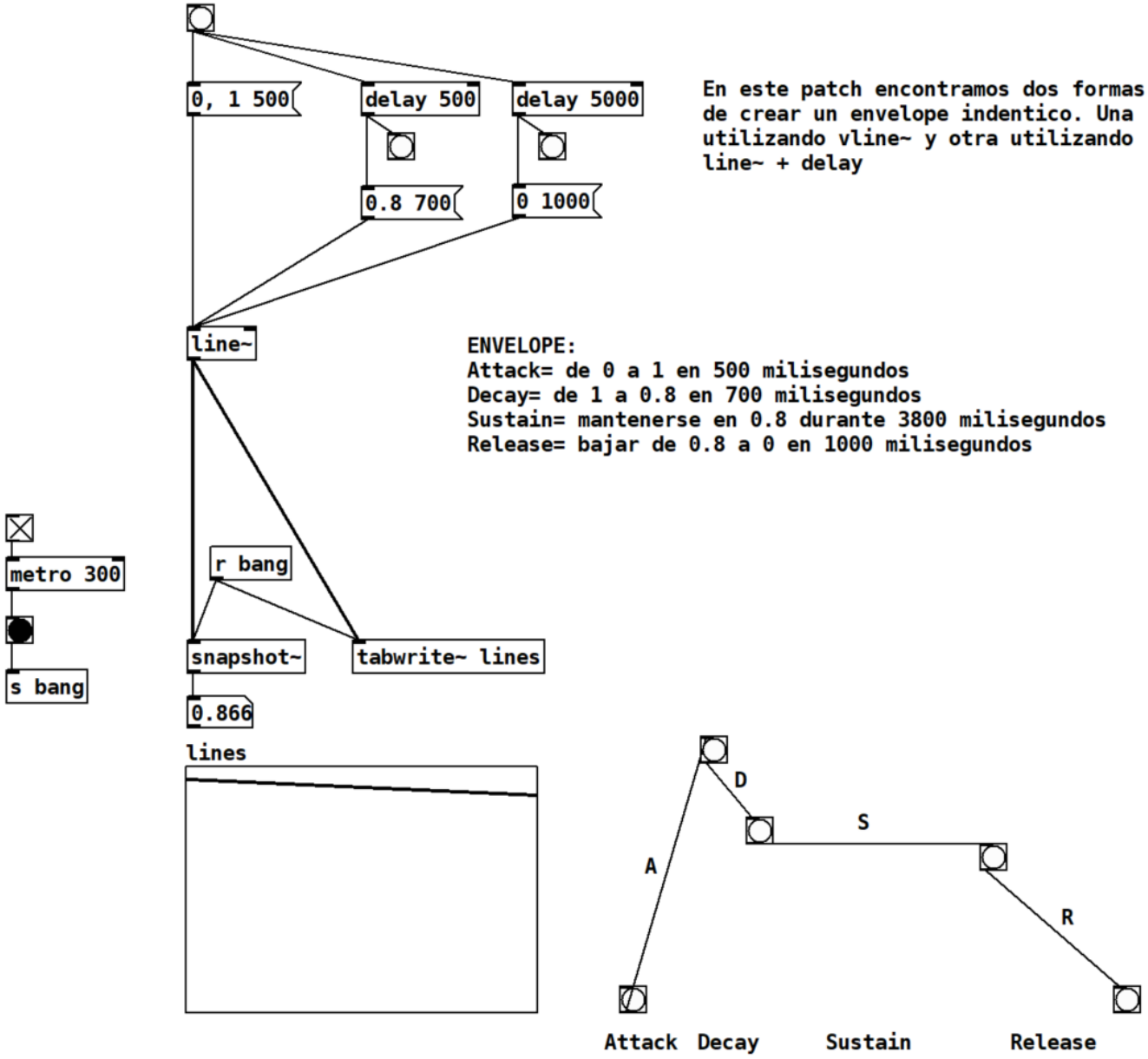


Figura 6. patch *line-ADSR.pd*

En la figura 4. vemos el patch *line-ADSR.pd*. Para comenzar el envelope activaremos el bang que se encuentra arriba. Este bang envía el primer mensaje y activa los dos delays. Cuando termina la **primera rampa (el ataque)** después de medio segundo (**500 milisegundos**) comienza la segunda rampa, como podéis ver el **tiempo que dura la primera rampa y el tiempo del primer delay es el mismo** para que justo en el momento en que termine la primera rampa comience la siguiente rampa, la **segunda rampa** que durara **700 milisegundos**. Cuando termine la segunda rampa, habrán pasado **500+700=1200** milisegundos desde el comienzo del envelope. Si quisiéramos que la tercera rampa comenzara inmediatamente después de terminar la segunda

configuraríamos el segundo delay con un tiempo de 1200 milisegundos, pero queremos que la señal se **mantenga constante** durante **3800 milisegundos**, asi crearemos la sección de **Sustain**. Para esto configuraremos el **segundo delay** con un tiempo superior a 1200 en nuestro caso **1200+3800=5000** ya que 3800 es el tiempo que queremos que dure el sustain. Tras 5000 milisegundos el segundo delay envía el bang que enviará el mensaje con la **tercera rampa** al line~, comienza el release que en nuestro caso durara **1000 milisegundos**.

> Ejercicio 1: ¿Cuál es la duración total del envelope en el ejemplo anterior?

> Ejercicio 2: crea un patch que genere un envelope con un ataque que vaya de 0.5 a 1 en 300 milisegundos, un decay que baje a 0.7 en 900 milisegundos. Un sustain de 2000 milisegundos y un decay que baje a 0 en 1100 milisegundos. ¿Cuál es la duración total del envelope que acabáis de construir?

Como habéis visto en el video de arriba, el tiempo que dura cada segmento es un parámetro clave para darle forma al sonido. Ahora, vamos a modificar el patch anterior para poder **controlar el tiempo de cada segmento**, y para ello vamos a introducir una utilidad de Pure data que nos va a permitir modificar el valor o los valores de un mensaje u objeto. Es el símbolo del dólar **$**.

$

Utilizaremos el símbolo **$** para remplazar un valor numérico o un símbolo en un mensaje u objeto. Es una forma de crear una **variable** que podamos modificar dentro de un mensaje u objeto. Acompañaremos $ con el número que indica la posición del valor en el mensaje u objeto, en ente caso comenzamos a contar en el 1. Si tenemos un mensaje con dos valores, el **primero** será **$1** y el **segundo $2**. El $1 y el $2 van a ser remplazados por los valores recibidos. Vamos a ver un ejemplo en el patch "variable-dollar-demo.pd":

Copyright 2026

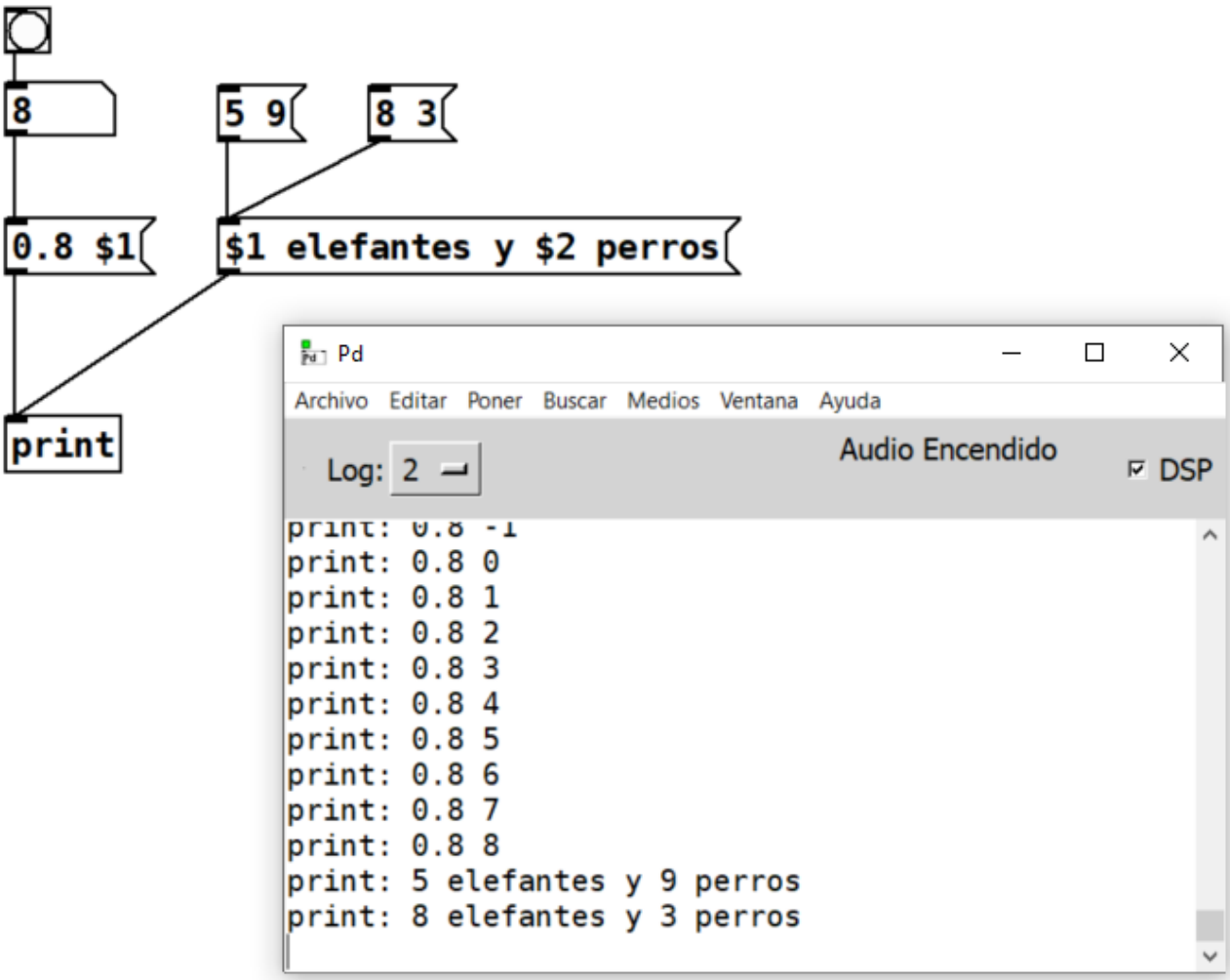


Figura 7. patch *variable-dollar-demo.pd*

Si os acordáis habíamos visto que en programación se empieza a contar en el 0, entonces, ¿porque el primer valor variable cuando utilizamos $ lo nombramos con 1? Es poque el **$0** tiene otra función, y lo utilizaremos para indicar que un **objeto** al cual le asignamos nosotros un nombre, como por ejemplo "tabwrite~", array, "send", "receive", ... **funcionan e interaccionan solo con objetos del patch en el que se encuentra**, es decir tiene un **alcance local**.

Esto que quiere decir y por qué nos va a ser util? Por ejemplo, teniendo dos patches abiertos con objetos "send" y "receive" nombrados con la misma etiqueta, "send go" y "receive go", lo que envíe el send del patch 1 va a llegar también al receive del patch 2, para evitar esto, colocaremos el $0 al comenzar el nombre de nuestra etiqueta: "send $0-go" y "receive $0-go", así, lo que envíe el "send $0-go" del patch 1 solo llegara al "receive $0-go"del patch 1. Como os podéis imaginar utilizaremos esto para objetos que no están conectados a traves de líneas, como el "tabwrite~" y la interfaz grafica array. En el siguiente capitulo veréis como esta utilizado en los patches.

Copyright 2026

Ahora que ya sabemos cómo introducir valores variables en mensaje, vamos a modificar el patch anterior para poder controlar el tiempo que dura el attack, el decay, el sustain y el release:

- La duración del **primer delay** va a ser la misma que la duración de la **primera rampa (Attack)**.
- La duración del **Decay** sera la duración de la **segunda rampa**.
- La duración del **segundo delay** sera la suma de la duración del **Attack más** la duración del **Decay más** el tiempo de duración del **Sustain**.
- La duración del **Release** sera la duración de la **tercera rampa**

Recordar que en el objeto suma el inlet derecho actualiza el valor de uno de los sumandos, pero al actualizar ese valor no se emite el resultado por la salida. El objeto + va a enviar el resultado solo cuando reciba un valor o bang en el inlet izquierdo, por eso cuando actualizamos el valor del inlet derecho enviamos también un bang en el inlet izquierdo.

Copyright 2026

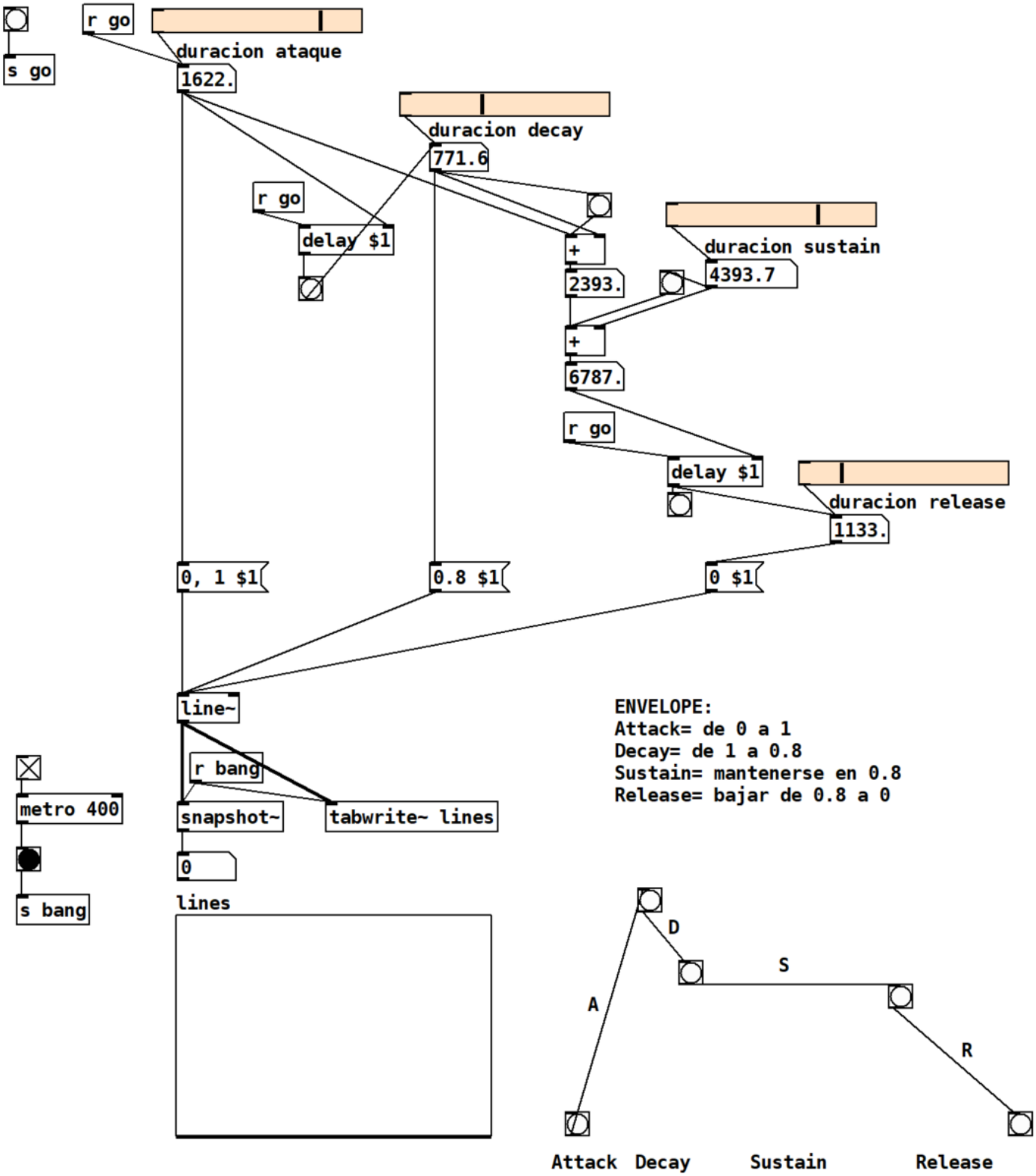


Figura 8. patch *line-ADSR-tiempos-variables.pd*

Vamos a ver ahora otra manera de construir estos envelopes en pure data utilizando el objeto "vline~".

vline~

Este objeto nos va a permitir construir envelopes **combinando rampas** de la misma forma que la combinación de delay y line~. De momento utilizaremos solo el **inlet izquierdo** para configurar los parámetros del vline~, que espera un **mensaje** con la información de las rampas que ha de realizar y cuando. La información de **cada rampa** estará **separada por una coma**, detrás de cada coma indicaremos el **valor de destino**, el **tiempo** que tardará en llegar al valor de destino y el **tiempo** que tardará esa rampa en comenzar **desde que empieza el envelope**. Si queremos configurar un valor inicial pondremos dicho valor al principio separado de una coma. El mensaje tendría la siguiente forma:

valor-inicial, valordestino-rampa1 tiempo-rampa1 delay-rampa1, valordestino-rampa-2 tiempo-rampa2 delay-rampa2

0, 1 500, 0.8 700 500

Rampa 1, Rampa 2

Para la primera rampa generalmente no pondremos delay porque al ser la primera comienza a la vez que se inicia el envelope. Y el delay de cada rampa es el tiempo desde que empieza el envelope hasta el momento en que queremos que commence esa rampa.

Vamos a ver ahora en el patch xxx como construir el mismo envelope que hicimos en el ejemplo de la figura 5, pero utilizando el objeto "**vline~**".

Copyright 2026

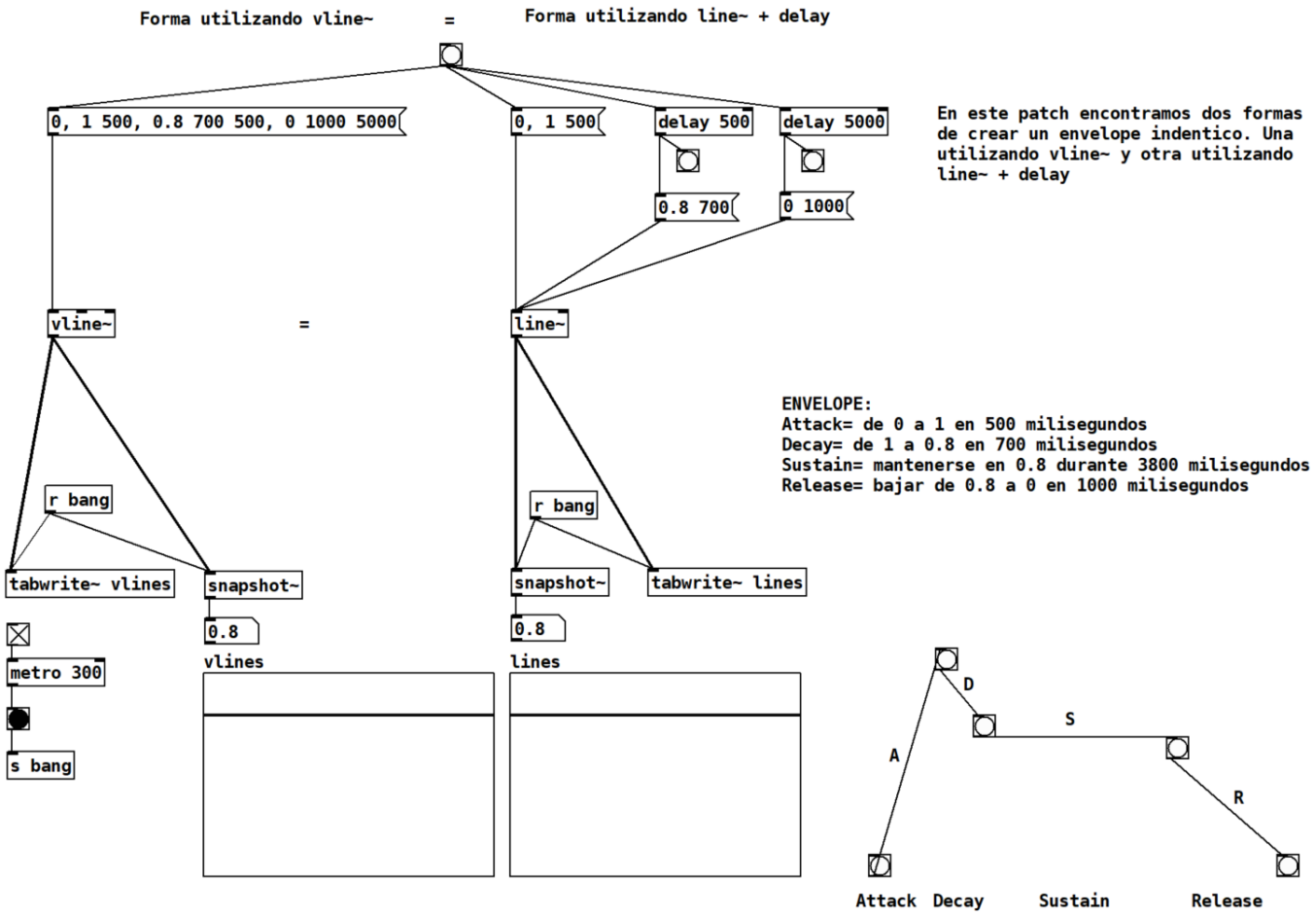


Figura 9. patch *line-vline-ADSR.pd*

> Ejercicio 3: Construye el mismo envelope que hiciste en el ejercicio 1 utilizando el objeto vline~

> Ejercicio 4: Modifica el patch en la Práctica 2, para ello, utiliza un **envelope ADSR** con el tiempo de decay variable que **regule el volumen** cada vez que se reproduzca una nota.

Figuras:

Figura 1. Tortuga deslizándose por una rampa. https://gifer.com/en/cih

Figura 2. ADSR envelope. https://thewolfsound.com/envelopes/

Figura 3. patch *delay-demo.pd*

Figura 4. Tortuga deslizándose por una rampa. https://gifer.com/en/cih

Figura 5. Combinación de varias rampas. https://giphy.com/gifs/southparkgifs-l0HluCIeReFkeWAQ8

Figura 5. patch *line-ADSR.pd*

Figura 6. patch *variable-dollar-demo.pd*

Figura 7. patch *line-ADSR-tiempos-variables.pd*

Figura 8. patch *line-vline-ADSR.pd*

Referencias:

Wilczek, J. (2022, Julio 3) *Envelopes in Sound Synthesis: The Ultimate Guide*. WolfSound. https://thewolfsound.com/envelopes/

Revision #1
Created 2023-02-17 11:30:16 CET by Julia del Río
Updated 2023-02-17 11:30:16 CET by Julia del Río

Copyright 2026