



TECNOLOGÍA DE ALTO NIVEL QUE INSPIRA CREATIVIDAD

Taller: Introducción a la Programación de Plug-ins de Audio



Agenda

Sobre nosotros

JUCE

Procesamiento de Audio con JUCE

Algoritmo de distorsión

Desarrollo de Plugin

Comunidad

Preguntas





Fernando Garcia

DSP Engineer | Audio Developer

Ingeniero en comunicaciones y electrónica

- Diseño de algoritmos.
- Implementación de estructuras de audio y procesamiento.
- Research and Developement





Jesús Valdez

Director de Software

Ingeniero en Producción Musical

- Dirección de proyectos en ECT.
- Desarrollo de back-end y front-end.
- Especialización en JUCE y desarrollo web.

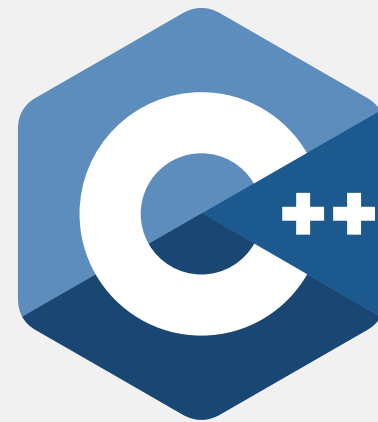


jsvaldezv



Herramientas

Integrated Development Environment (IDE)

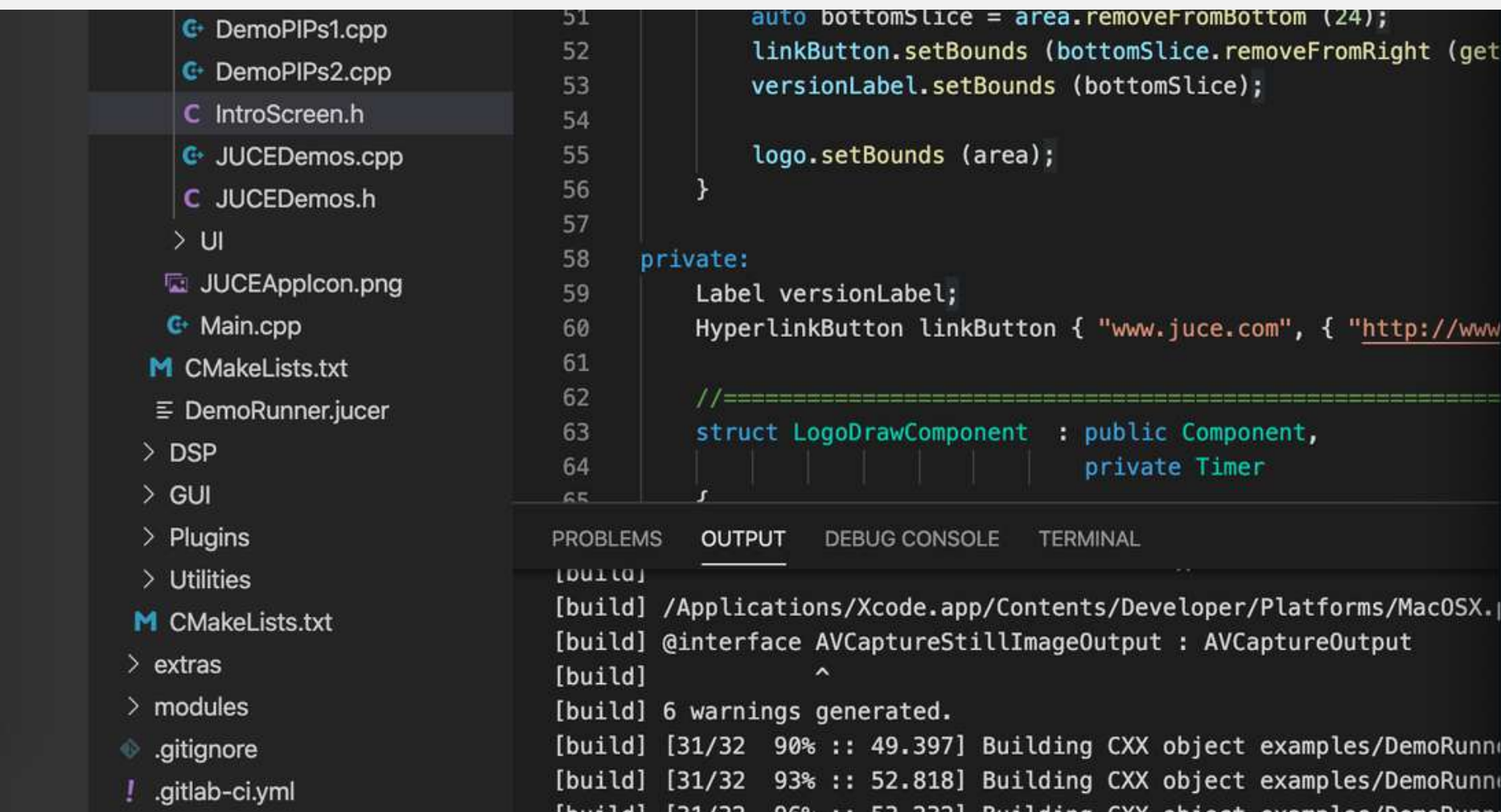


JUCE



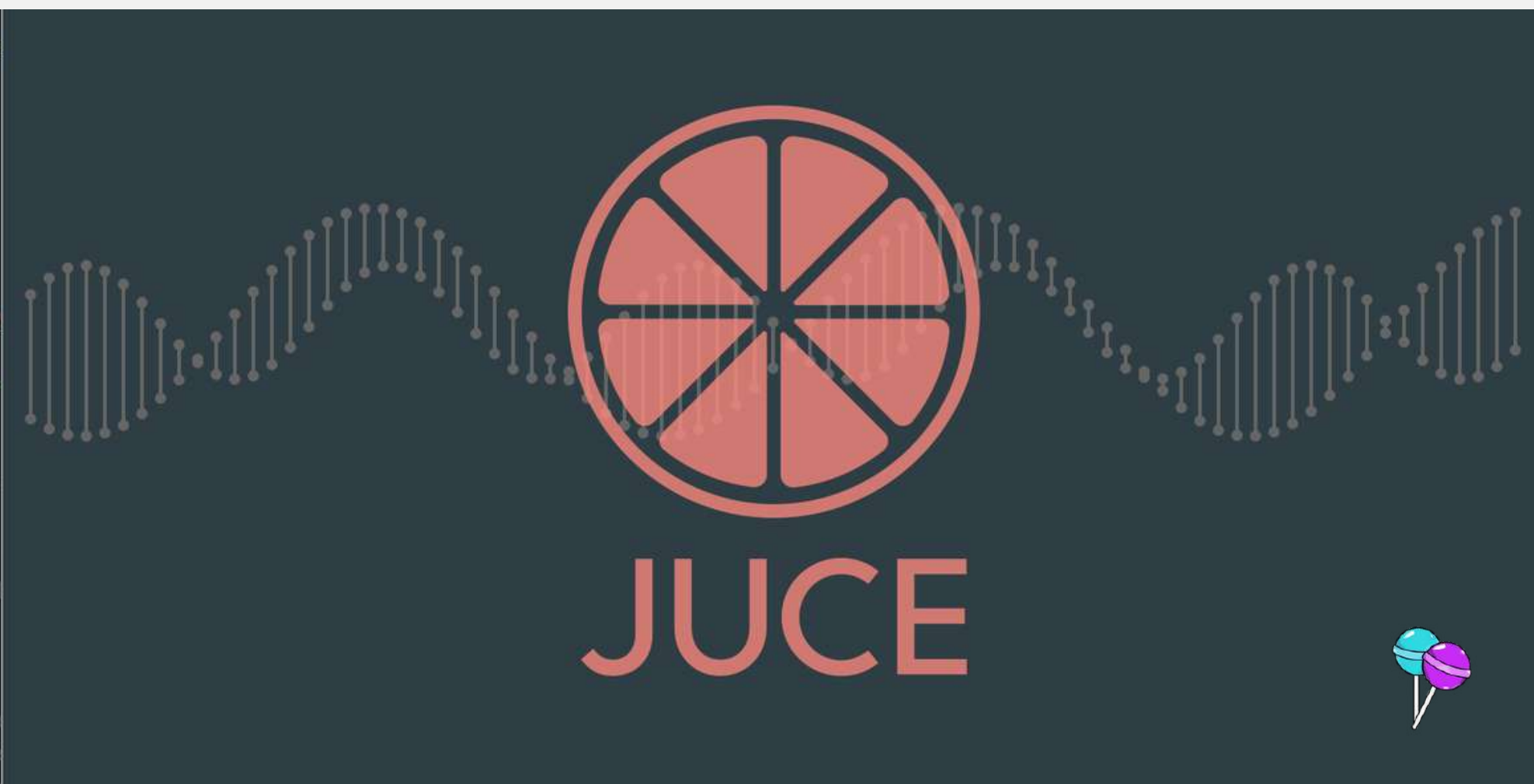
JUCE

- Framework de C++ enfocado al desarrollo de plug-ins y aplicaciones de audio.
- Principal framework de audio usado a nivel mundial.
- Multiplataforma.
- Para MacOS, Windows, Linux, IOS y Android.



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer lists files like DemoPIPs1.cpp, DemoPIPs2.cpp, IntroScreen.h, JUCEDemos.cpp, JUCEDemos.h, UI, JUCEAppIcon.png, Main.cpp, CMakeLists.txt, DemoRunner.jucer, DSP, GUI, Plugins, Utilities, CMakeLists.txt, extras, modules, .gitignore, and .gitlab-ci.yml. The code editor shows a C++ snippet with line numbers 51 to 65. The code includes a function call to removeFromBottom, setBounds, and a private section with a Label and HyperlinkButton. A comment line is also present.

```
51 auto bottomSlice = area.removeFromBottom (24);
52 linkButton.setBounds (bottomSlice.removeFromRight (get
53 versionLabel.setBounds (bottomSlice);
54
55 logo.setBounds (area);
56 }
57
58 private:
59 Label versionLabel;
60 HyperlinkButton linkButton { "www.juce.com", { "http://www
61
62 //=====
63 struct LogoDrawComponent : public Component,
64 private Timer
65 }
```



Uso actual en la industria

- Desarrollo de plug-ins de audio
- Desarrollo de aplicaciones mobiles
- Desarrollo de librerías/modulos de C++
- Desarrollo de DAWs



Vital Synth



DAW Waveform

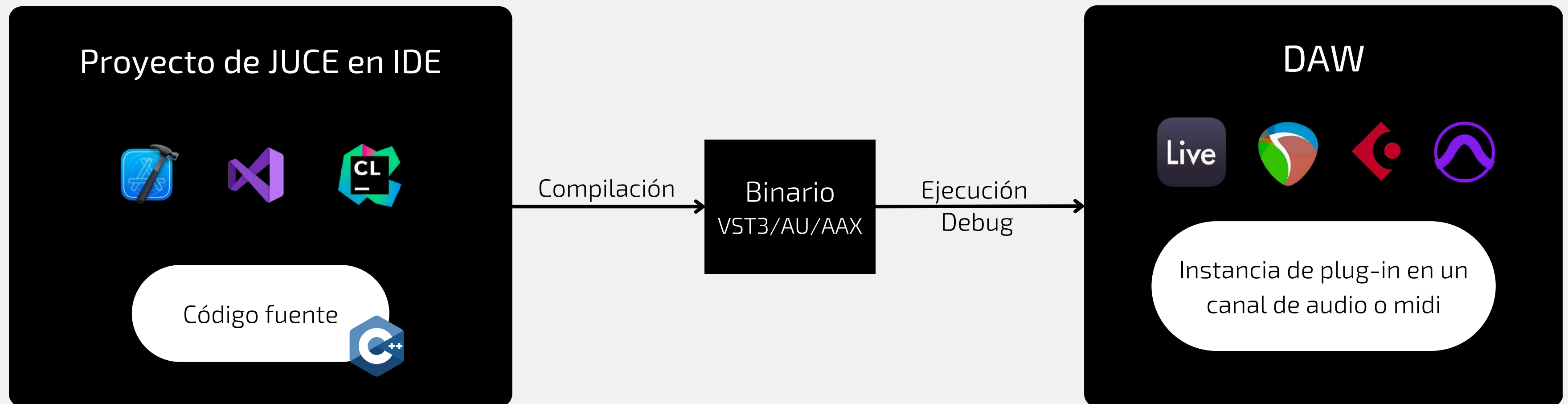


Output Plug-ins

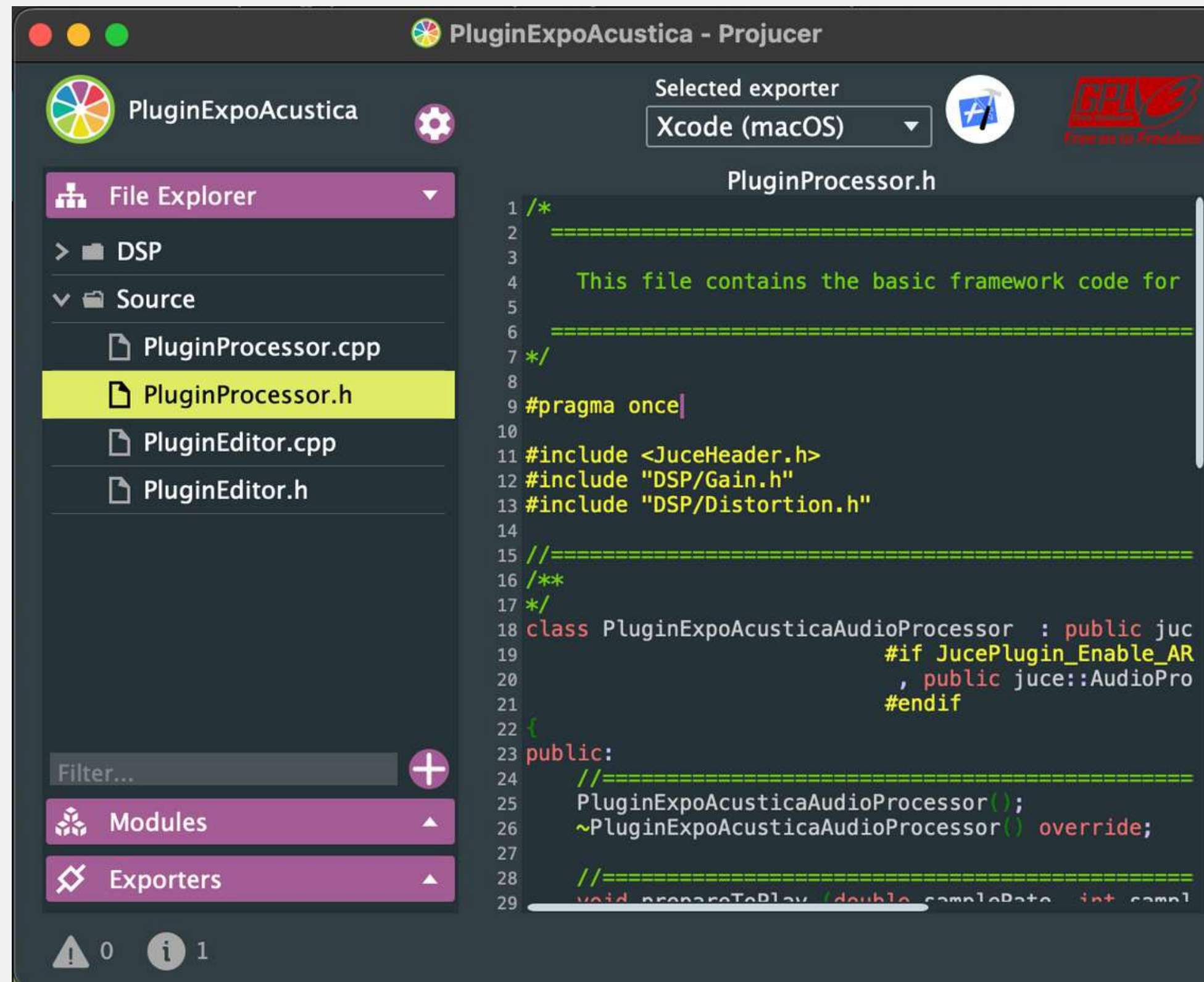


Arquitectura de desarrollo de plug-ins en JUCE

- Se procesa por audio buffers en "tiempo real"
- Algoritmos:
 - Fórmula directa / Ecuaciones de diferencias.
 - Módulos y/o librerías de DSP internas de JUCE o de la comunidad.



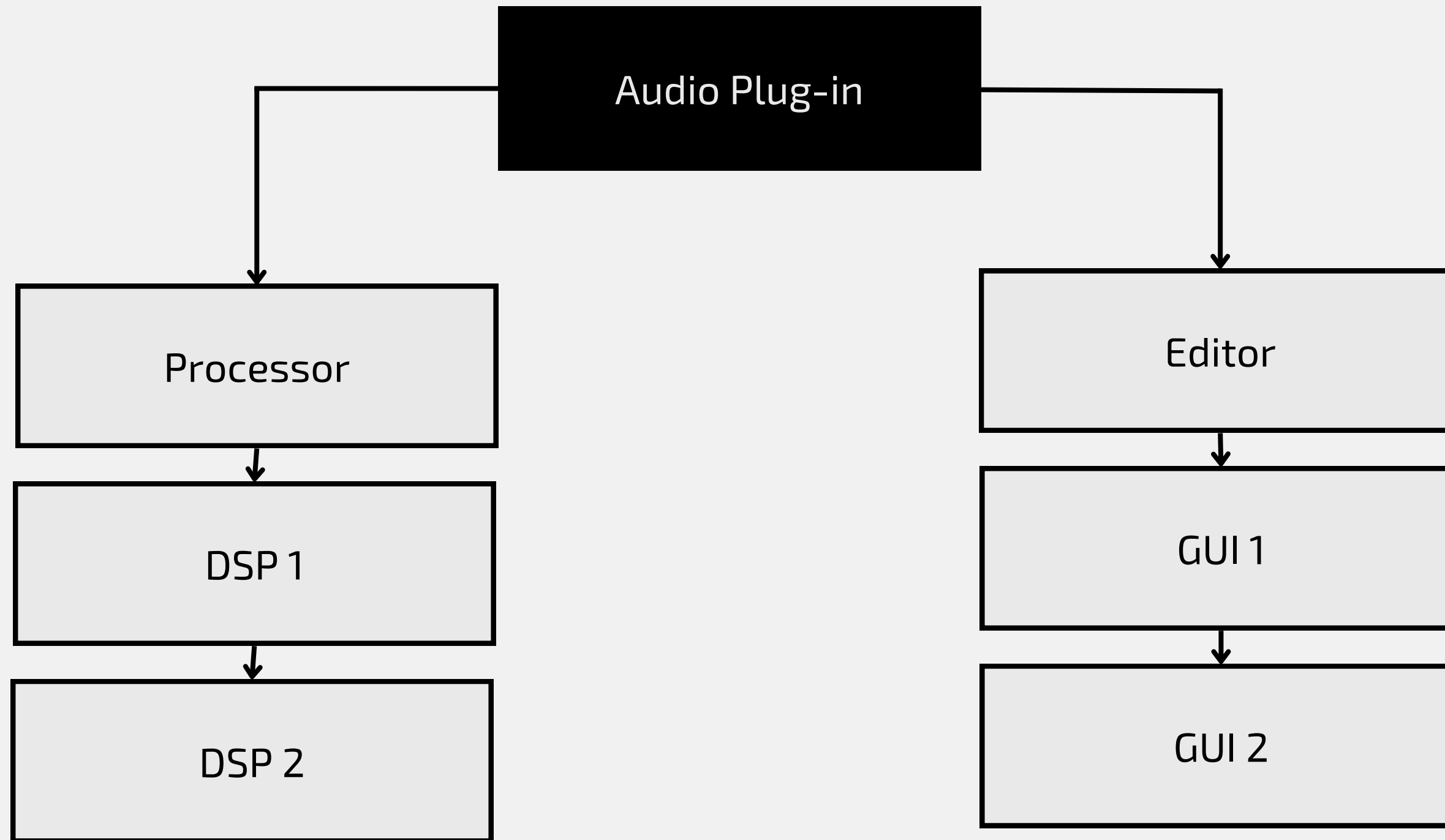
Proyecto de JUCE



Projucer → Basic Audio Plug-In



Estructura de JUCE



Partes de un plugin en JUCE

Processor

- Procesamiento de audio
- Corazón del plugin
- Programación de cualquier tipo de algoritmos para audio
- `processBlock()`

Editor

- Elemento de interfaz gráfica de usuario (GUI)
- Botones, faders, sliders, imágenes, etc.



Procesamiento de Audio con JUCE

Conceptos importantes de DSP y Audio Digital:

- Frecuencia de muestreo (Sampling Frequency)
- Profundidad de bits (BitDepth)
- Procesamiento por buffers de audio
 - Buffer Size
- Algoritmos**

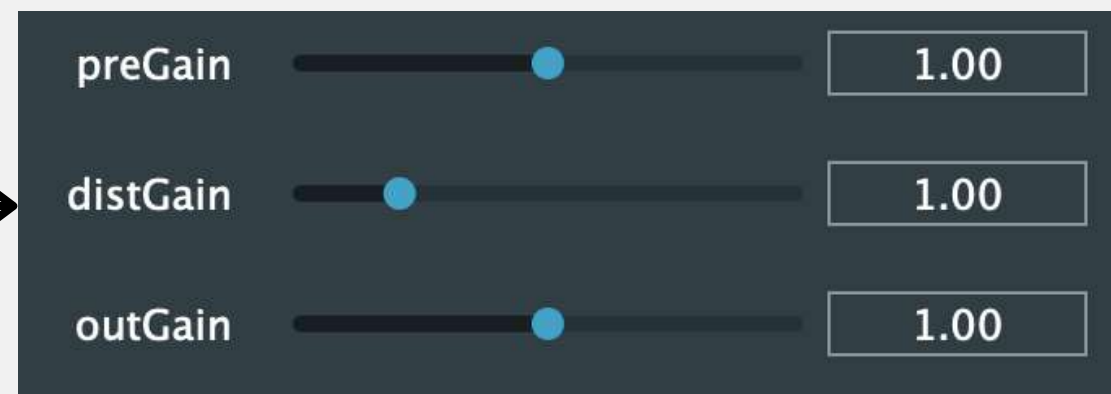
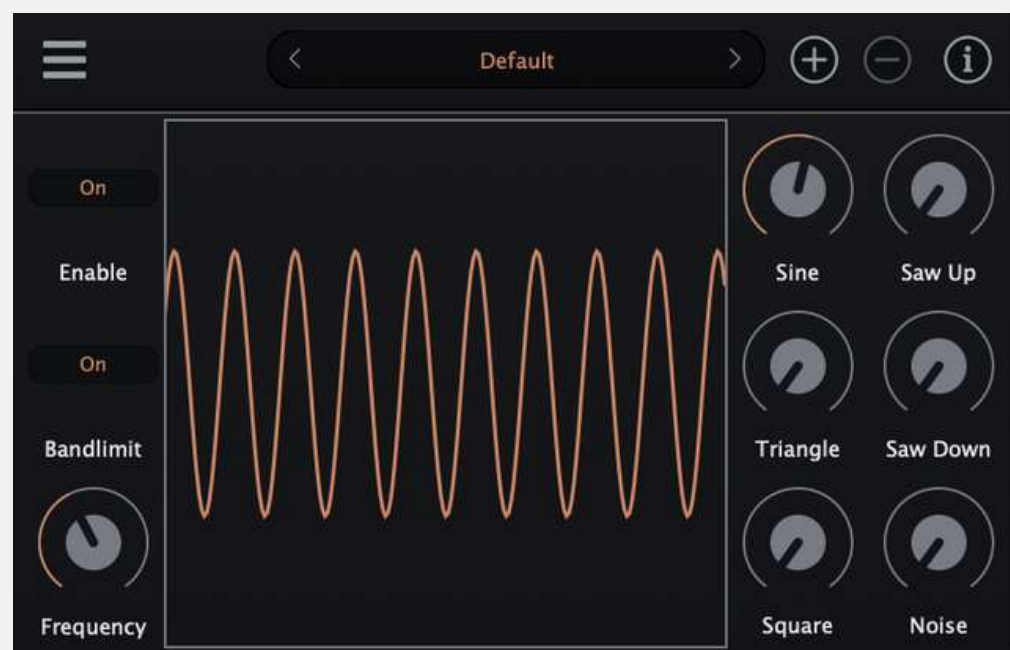


** Implementados desde cero o con librerías

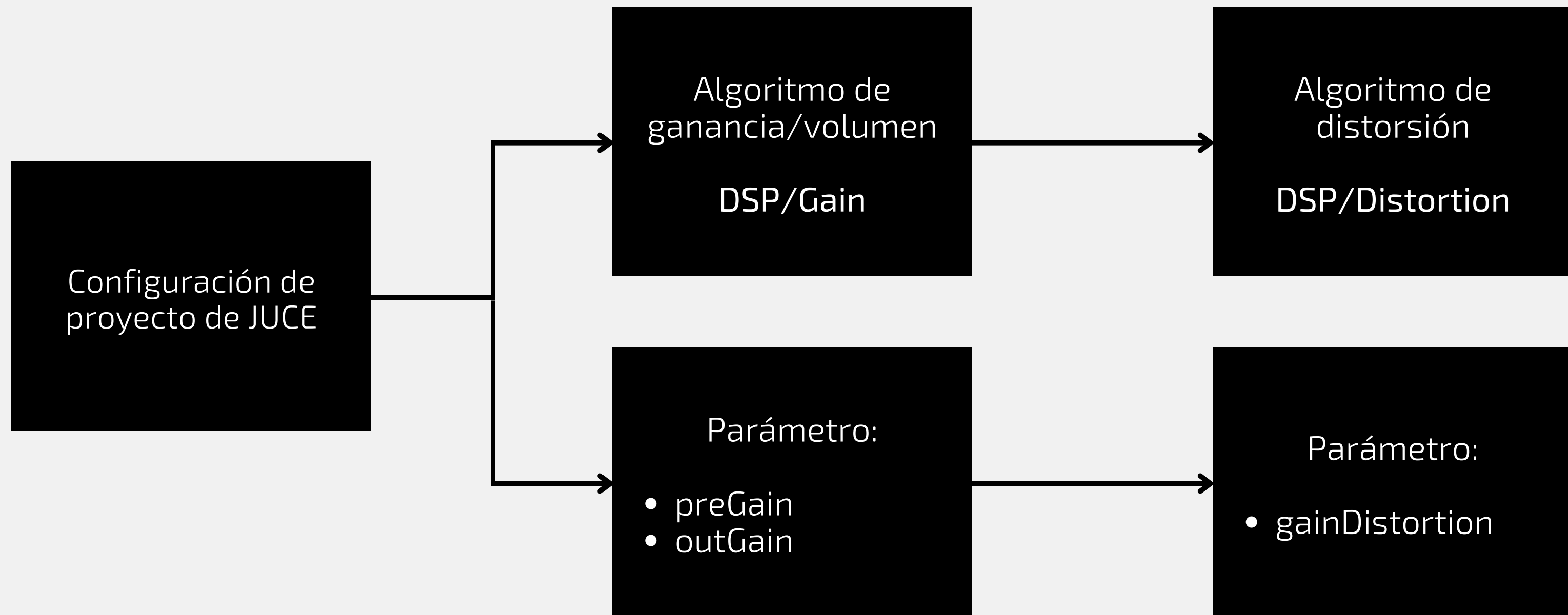


Práctica:

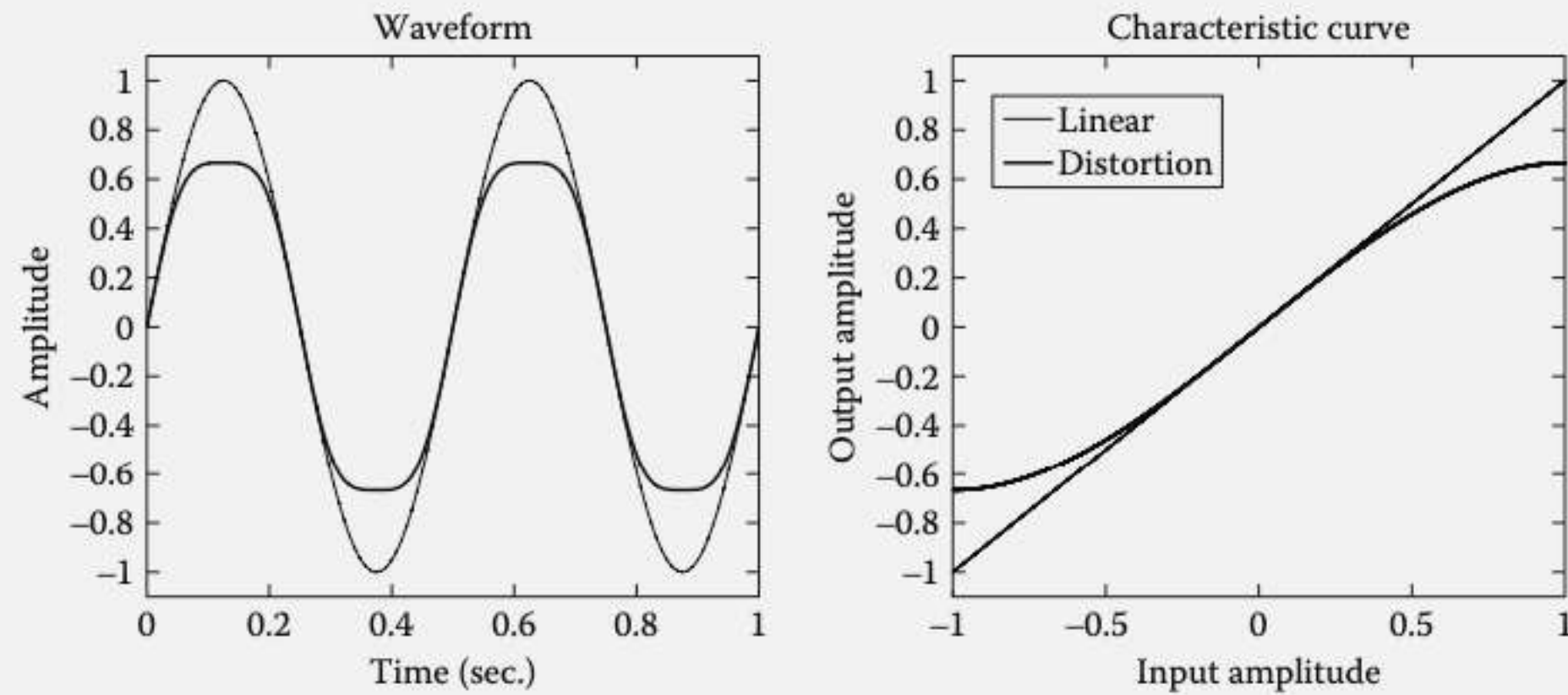
¡Programemos un plugin de distorsión!



Proceso



Distorsión



Soft Clipping

$$y[n] = x[n] - \frac{x[n]^3}{3}$$



Otras ecuaciones de distorsión

Half Wave Rectifier	HWR	$y = 0.5(x + x)$
Full-Wave Rectifier	FWR	$y = x $
Square Law	SQR	$y = x^2$
Absolute Square Root	ASQRT	$y = \sqrt{ x }$

- Pirkle, W. (2019). Designing audio effect plugins in C++: For AAX, AU, and VST3 with DSP theory (2a ed.). Routledge.





```
1 void Distortion::process(juce::AudioBuffer<float> &inBuffer, float distortionGainValue)
2 {
3     for (int channel = 0; channel < inBuffer.getNumChannels(); channel++)
4     {
5         for (int i = 0; i < inBuffer.getNumSamples(); i++)
6         {
7             auto inSample = inBuffer.getSample (channel, i);
8             // Soft Clip Algorithm
9             float gainSample = distortionGainValue * inSample;
10            float outSample = gainSample - (powf(gainSample, 3.0f) / 3.0f) ;
11            inBuffer.setSample(channel, i, outSample);
12        }
13    }
14 }
```



Procesamiento por Buffers de audio

Stream de audio



Buffer

Buffer

Buffer



Procesamiento por Buffers de audio

Stream de audio



Buffer

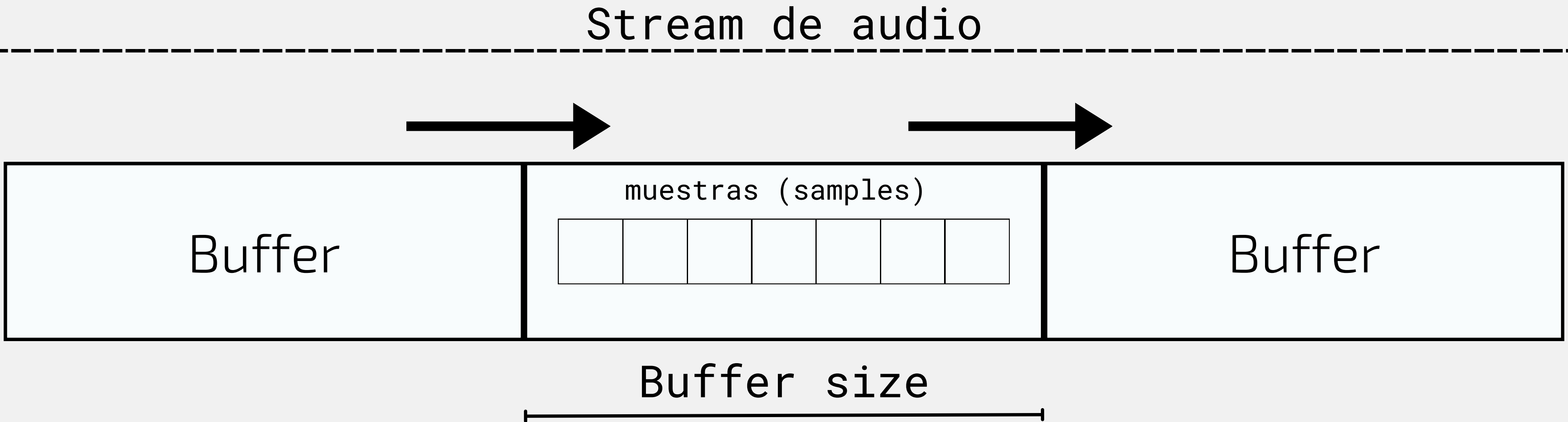
muestras (samples)



Buffer



Procesamiento por Buffers de audio

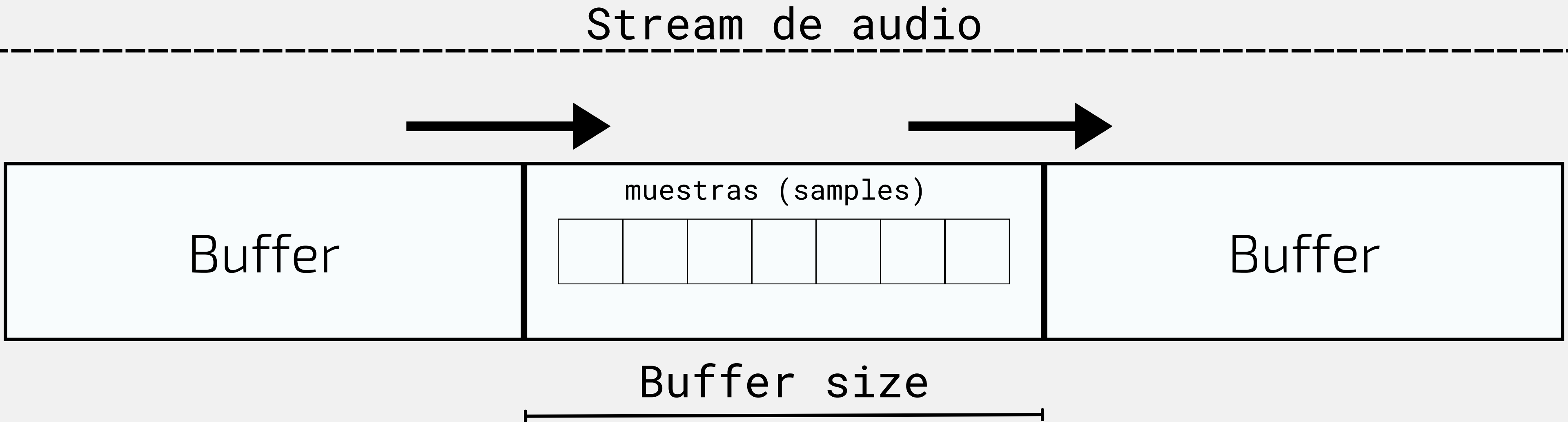


Ejemplos de
tamaño de
buffer →

256
512
1,024
2,048



Procesamiento por Buffers de audio



Ejemplos de tamaño de buffer

256
512
1,024
2,048

→ 11.6 ms (44.1 kHz)



Procesamiento por Buffers de audio

Stream de audio



Buffer

muestras (samples)

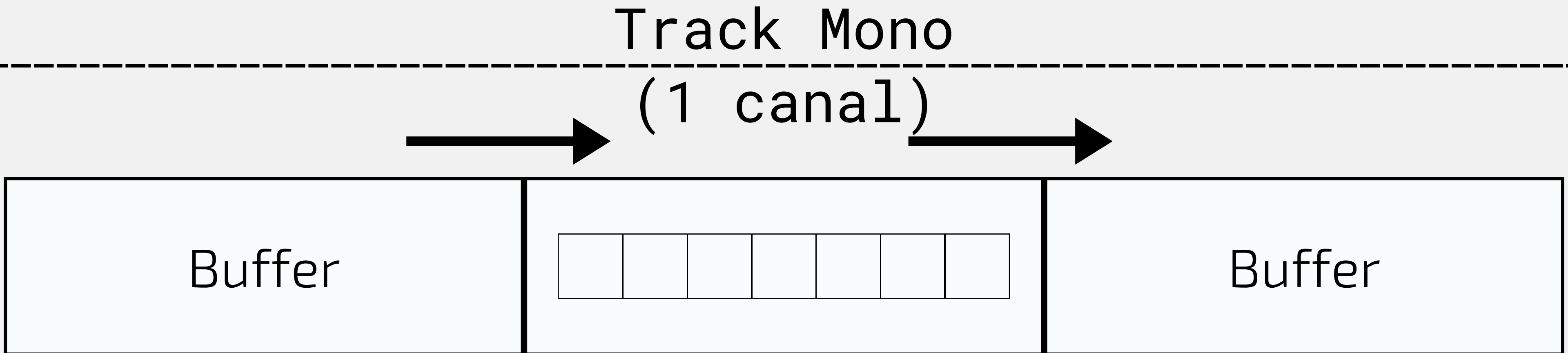
0.23	-0.12	-0.23	0.0	0.24	0.43	0.99
------	-------	-------	-----	------	------	------

Buffer

Valores que pueden tomar una muestra
[-1, 1]
(número flotante)



Procesamiento por Buffers de audio

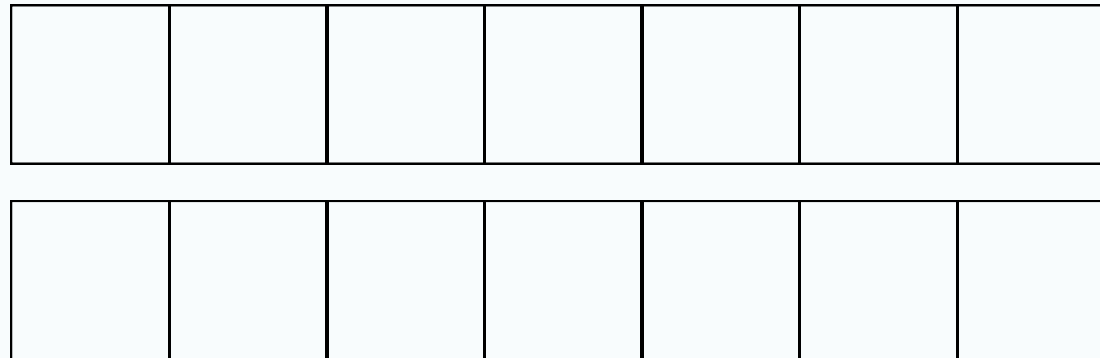


Procesamiento por Buffers de audio

Track Estéreo
(2 canales)



Buffer



Buffer

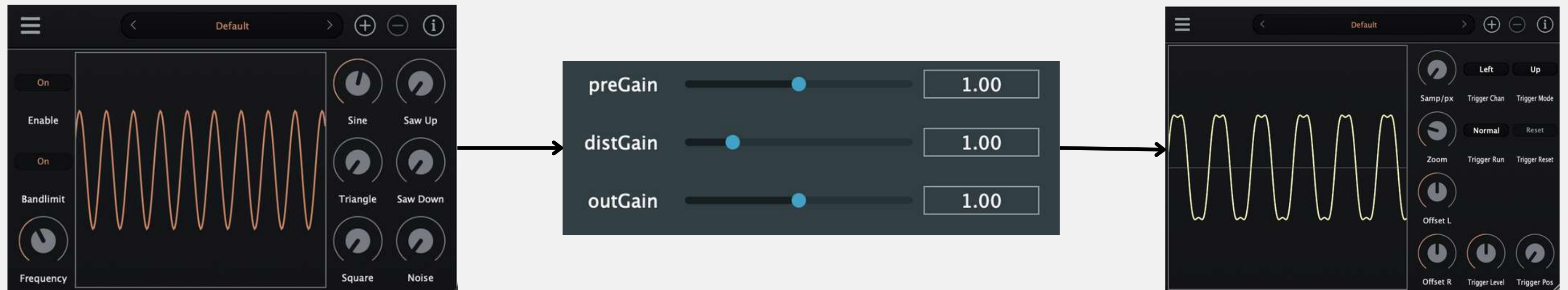




```
1 void Distortion::process(juce::AudioBuffer<float> &inBuffer, float distortionGainValue)
2 {
3     for (int channel = 0; channel < inBuffer.getNumChannels(); channel++)
4     {
5         for (int i = 0; i < inBuffer.getNumSamples(); i++)
6         {
7             auto inSample = inBuffer.getSample (channel, i);
8             // Soft Clip Algorithm
9             float gainSample = distortionGainValue * inSample;
10            float outSample = gainSample - (powf(gainSample, 3.0f) / 3.0f) ;
11            inBuffer.setSample(channel, i, outSample);
12        }
13    }
14 }
```



Coding Time



Repositorio de Github

<https://github.com/Ear-Candy-Technologies/intro-audio-plugins-expoacustica2023>



COMUNIDAD



AMARANTH

¿QUÉ ES?

Sintetizador Open Source
desarrollado con JUCE

¿PUEDO PARTICIPAR?

Cualquier persona es libre de
participar y desarrollar nuevos
features en el.

```
7 void DryWet::process (juce::AudioSampleBuffer& juce::A
8                               juce::A
9                               float d
10 {
11     drywetValue = drywetValue
12
13     for (int channel = 0; cha
14     {
15         for (int i = 0; i < d
16         {
17             // Wet sample
18             float wet = wetBu
19
20             // Dry sample
21             float dry = dryBu
22
23             // DryWet process
24             float out = dry *
25
26             // Output
27             wetBuffer.setSamp
28         }
29     }
30 }
```



EAR CANDY MEETINGS

¿QUÉ SON?

Livestreams en nuestro canal de YouTube y Facebook donde expertos de la industria platican sobre un tema relacionado a la programación de audio.

¿CADA CUANTO?

Se llevan a cabo el último miércoles de cada mes.

EAR CANDY MEETINGS

by Ear Candy Technologies



RODRIGO UF
StageWave

“Librería de audio
para Android



Ear Candy Technologies



Ear Candy Technologies



¿Preguntas?





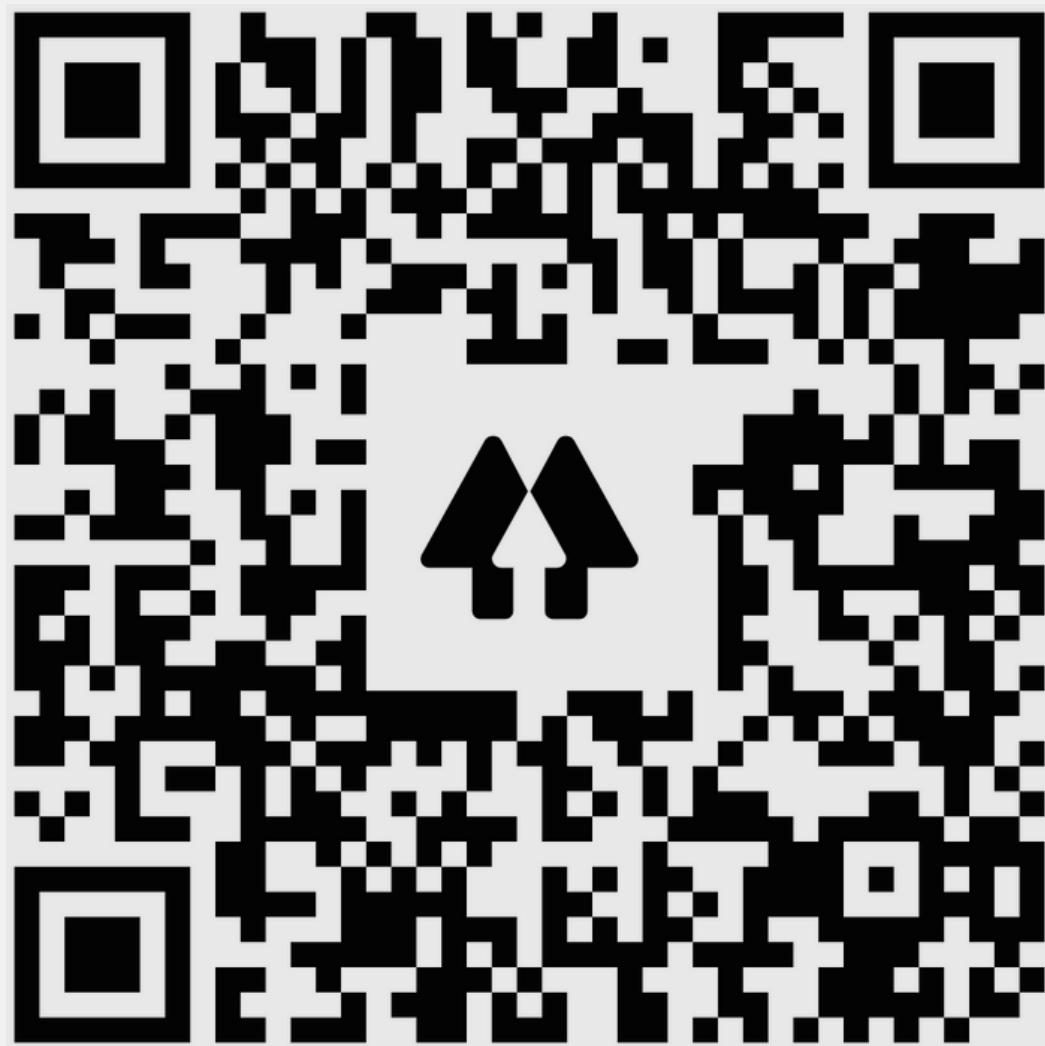
PHONOGRain

50% de descuento en
Phonograin con el código:

EXPO_ACUSTICA_23



¡Gracias!



www.earcandytech.com



Ear Candy Technologies



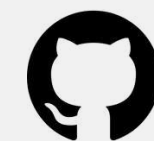
Ear Candy Technologies



earcandytech



jsvaldezv



fergarciadlc