SISTEMA AVANZADO DE VISUALIZACIÓN DE AUDIO

CRISTIAN JUNIOR CUNURANA CALDERON

INTRODUCCIÓN

El presente proyecto implementa un visualizador de audio avanzado desarrollado en C++ utilizando la librería SFML para el renderizado gráfico y FFTW para el análisis de frecuencias.

El objetivo principal es transformar la señal de audio en tiempo real en una representación visual dinámica, fluida y atractiva, aprovechando distintos efectos como curvas de ondas, espectros circulares, partículas y una paleta de colores tipo neon/cyberpunk.

MUSIC GENRE

El proyecto está desarrollado en C++, un lenguaje ampliamente utilizado en aplicaciones donde el rendimiento es crítico, como procesamiento de audio, gráficos, videojuegos o sistemas embebidos. Su eficiencia, control de memoria y capacidad para trabajar cerca del hardware lo convierten en la opción ideal para un visualizador de señales y efectos gráficos en tiempo real.

Para la configuración, compilación y organización del proyecto se utiliza CMake, una de las herramientas más usadas actualmente en proyectos profesionales de C++.

SFML es la librería central del proyecto y se utiliza tanto para el audio como para los gráficos. Con sf::SoundBuffer y sf::Sound obtenemos las muestras del sonido en tiempo real para generar amplitudes, intensidades y el espectro. En la parte visual, usando sf::RenderWindow y sf::VertexArray, SFML dibuja las ondas, barras y efectos sincronizados con el audio. Su integración simple de audio + gráficos la hace ideal para un visualizador en tiempo real.

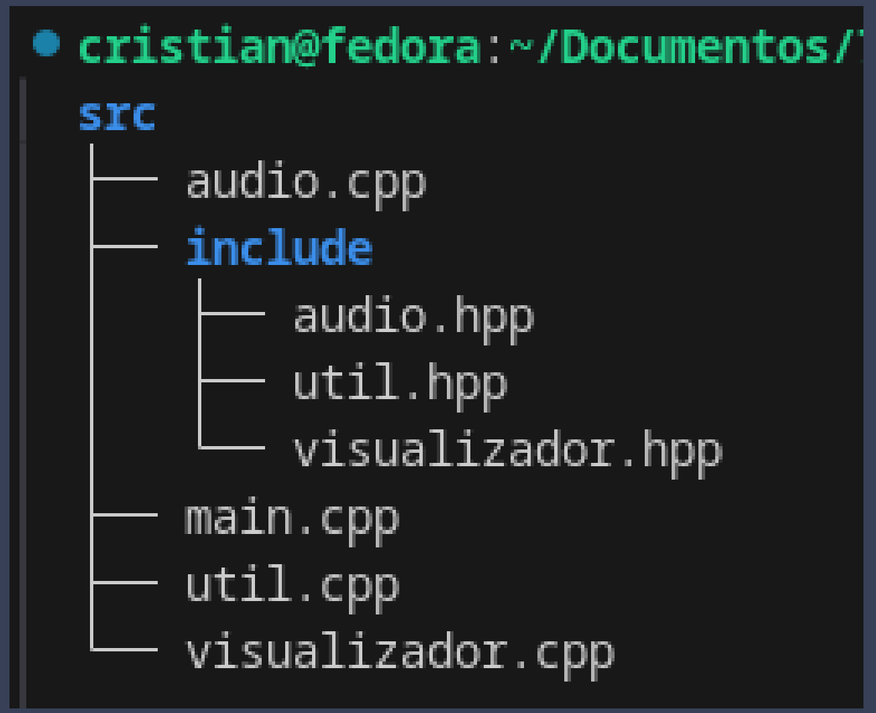
3. ARQUITECTURA GENERAL

Clases principales:

- Visualizador
- AudioManager
- Particula
- SFML RenderWindow

Flujo:

1. Cargar buffer de audio
2. Extraer ventana de samples
3. Ejecutar FFT
4. Procesar frecuencias → alturas
5. Actualizar ondas, partículas y colores
6. Dibujar en pantalla



4. ANÁLISIS DE FRECUENCIAS (FFT)

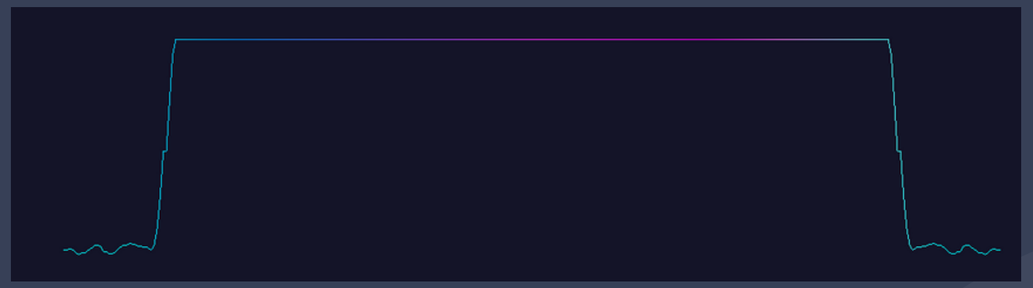
Función clave: analizarFrecuencias()

- Usa FFTW3 para transformar 2048 muestras.
- Mezcla estéreo → mono para análisis.
- Distribución calibrada para sensibilidad:
  - Bajos → ultra sensibles
  - Medios → sensibles
  - Agudos → controlados
- Aplicación de `log1p` para manejo dinámico
- Boost adicional para realzar graves/medios

PROCESAMIENTO DE ONDAS

Función: actualizarOndas()
Incluye:

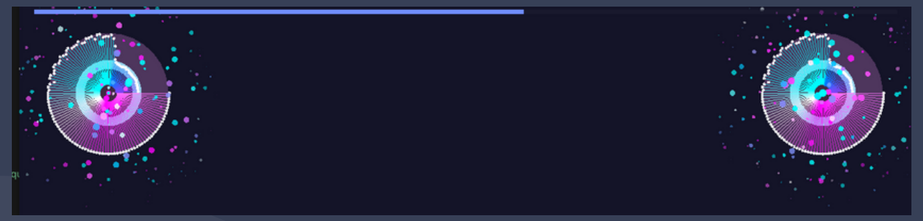
- 4 capas de ondas
- Espectro centrado (bajos al centro, agudos a extremos)
- Curva suavizada con interpolación
- Línea base estable
- Colores neón dinámicos tipo “cyberpunk”



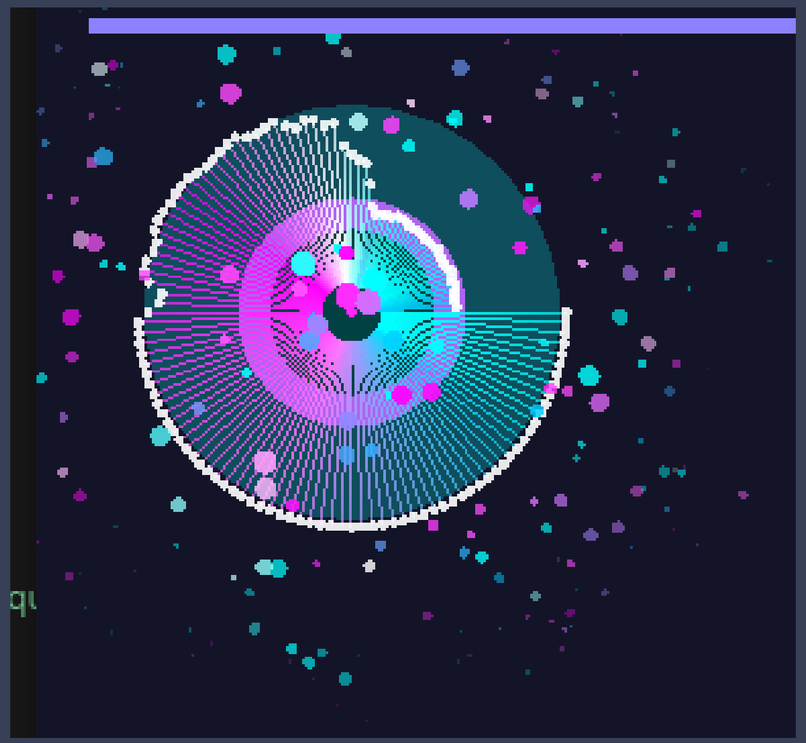
ECUALIZADOR CIRCULAR

Función: dibujarVisualizadorCircular()
Características:

- Dos círculos: izquierda y derecha
- Círculo central pulsante con el ritmo
- Barras ultra cortas (mini spikes)
- Glow dinámico
- Partícula en punta con threshold
- Arcoíris animado



EFECTO DE PARTÍCULAS



Funciones:

- crearParticulas()
- actualizarParticulas()
- dibujarEfectosParticulas()

Características:

- Nacen desde ambos círculos
- Dirección aleatoria
- Vida limitada con fade-out
- Color basado en paleta neón
- Reacción a intensidad promedio del audio


GRACIAS