

# **SDL Mixer**

[https://www.libsdl.org/projects/SDL\\_mixer](https://www.libsdl.org/projects/SDL_mixer)

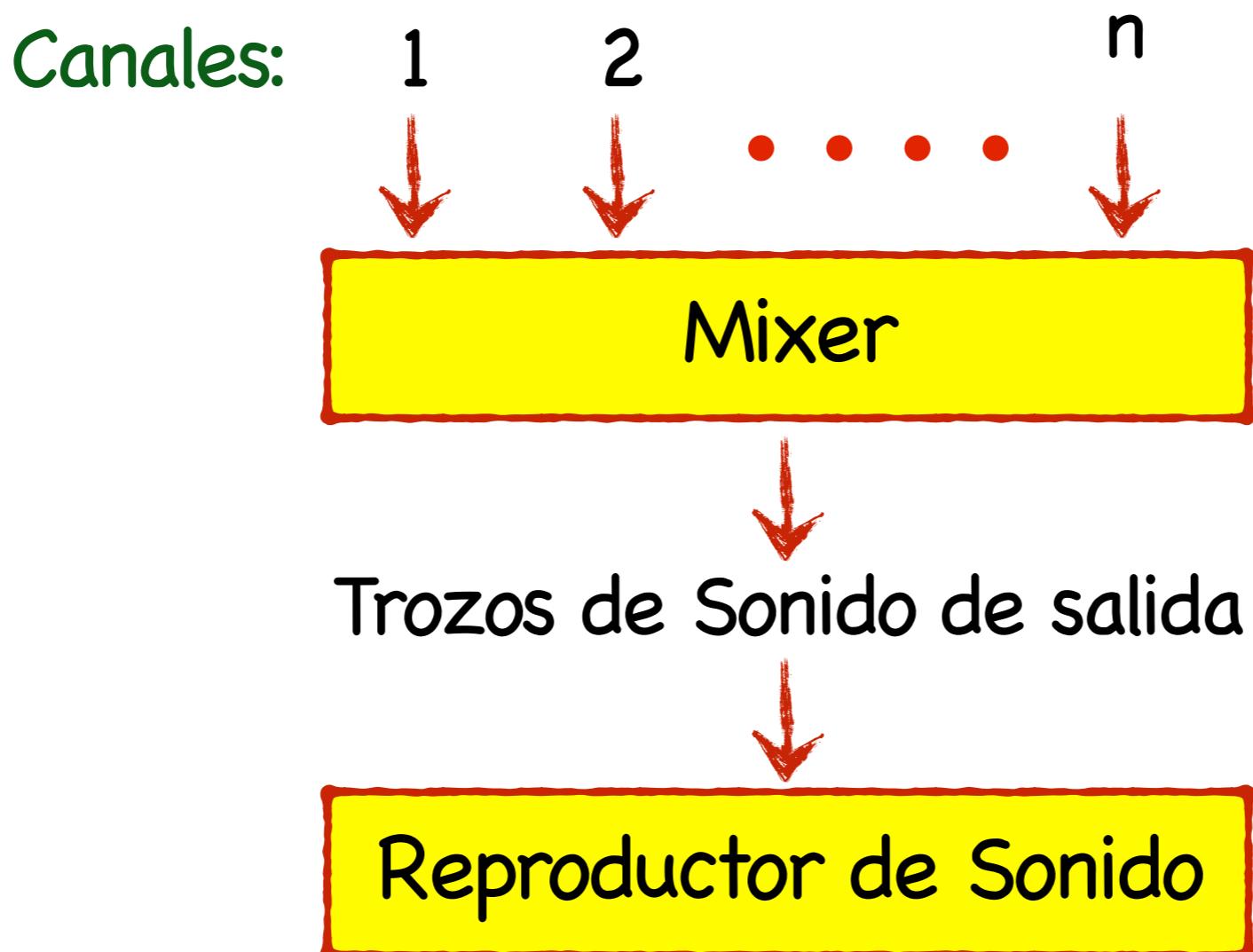
TPV  
Samir Genaim

# ¿Qué es SDL Mixer?

- ◆ SDL\_Mixer es una librería para reproducir sonido. SDL ya tiene una API para reproducir sonido, pero con SDL\_Mixer es mucho más fácil ...
- ◆ SDL\_Mixer permite reproducir dos tipos de sonido:
  - **Effects**: efectos de sonidos que se pueden reproducir en distintos canales y SDL\_Mixer las mezcla para que todos los sonido se reproduzcan a la vez ...
  - **Music**: se puede reproducir sólo un flojo de sonido (sólo un archivo), se maneja de manera separada de los canales pero se reproduce a la vez con los efectos de sonido ...
- ◆ La diferencia entre **Effects** y **Music** es en la calidad del sonido. Para los efectos no necesitamos alta calidad pero para la música sí ...
- ◆ Vamos a ver lo básico, más en la documentación ...

# Mixer: Esquema General

Sonidos de entrada, uno en cada canal. Se puede cambiar el volumen de cada canal, etc.



# Como Usar SDL\_Mixer

```
SDL_Init(...);
```

En los flags de `SDL_Init` hay que incluir `SDL_INIT_AUDIO`

```
// ...
```

```
// Initialize SDL_Mixer
```

Primero hay que inicializar `SDL_Mixer` llamando a `Mix_Init` y `Mix_OpenAudio`

```
// ...
```

```
// generate sounds, etc.
```

Reproducir sonido, etc.

```
// ...
```

```
// Finalize SDL_Mixer  
// ...
```

Al final hay que finalizar `SDL_Mixer` para liberar recursos, etc.

```
SDL_Quit();
```

# Mix\_Init y Mix\_Quit

```
Mix_Init( MIX_INIT_MP3 | MIX_INIT_OGG | ... );
```

Carga el soporte según lo indicado por los flags. Se puede llamar varias veces, no sólo al principio, no carga lo que está cargado ya. Bastaría sólo una llamada a Mix\_Quit a final.

Los posibles valores dependen de la versión usada, en 2.0 son  
MIX\_INIT\_FLAC, MIX\_INIT\_MOD, MIX\_INIT\_MP3,  
MIX\_INIT\_OGG y MIX\_INIT\_MID.

# Sound Effects

# Mix\_OpenAudio y Mix\_CloseAudio

Después de `Mix_Init` hay que configurar el mixer usando `Mix_OpenAudio`. Se puede llamar a `Mix_OpenAudio` varias veces para cambiar la configuración (para cambiar el formato hay que llamar a `Mix_CloseAudio()` antes)

```
Mix_OpenAudio(22050, MIX_DEFAULT_FORMAT, 2, 2048);
```

Frecuencia del sonido generado por el mixer.  
Normalmente se usan 44100 o 22050, el primero consume mucho más CPU

1 para MONO y  
2 para STEREO

El formato del sonido generado por el mixer: 8bits, 16bit, signed, unsigned, etc. Afecta la calidad, ver documentación para todos los valores ...

El mixer produce chunks (trozos) de sonido mezclando los flojos de los canales. Ese parámetro indica el tamaño de esos chunks. Brevemente, esos chunks se pasan a otras partes para reproducir el sonido, etc., así que ese parámetro afecta la frecuencia de llamadas a esas partes ...

# Fijar el Número de Canales

```
int Mix_AllocateChannels(int numchans);
```



Fijar el numero de canales, devuelve el numero de canales creado. Por defecto hay 8 canales ...

# Cargar un Archivo de Sonido

Carga el archivo de sonido (sólo una vez, se puede usar varias veces después). Devuelve null en caso de error.

```
Mix_Chunk *chunk = Mix_LoadWAV("GunShot.wav");
```

```
//...  
// Usar chunk para reproducir de sonido en un canal, etc,  
// ...
```

```
Mix_FreeChunk(chunk);
```

Al final liberar la memoria correspondiente, etc.

# Reproducir Efectos de Sonido

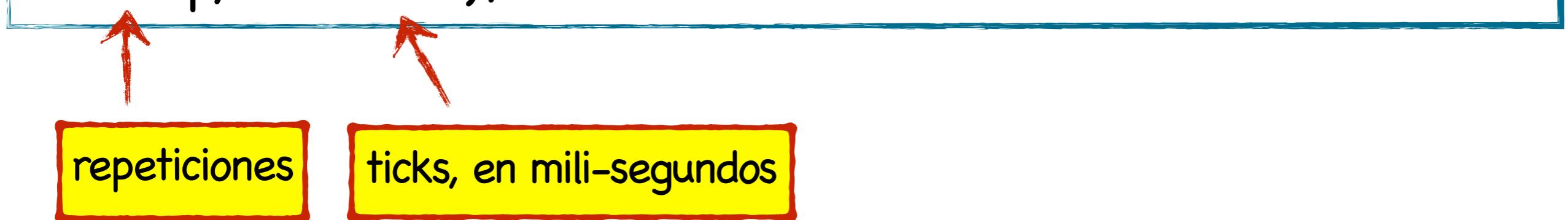
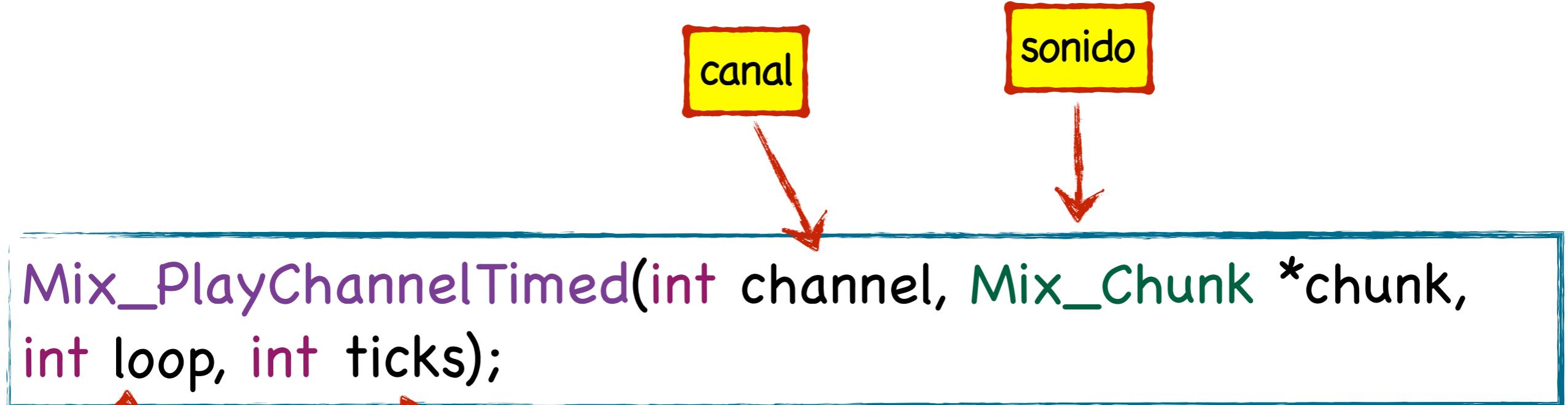


```
Mix_PlayChannel(int channel, Mix_Chunk *chunk, int loop);
```

Reproduce el sonido correspondiente en un canal específico. El primer parámetro es el canal (-1 busca el primer canal libre), el segundo es el sonido, el tercero es el numero de repeticiones (lo reproduce  $1+loop$  veces, -1 para reproducir infinitamente).

La llamada devuelve el canal en el que está reproduciendo chunk, -1 en caso de error.

# Reproducir con Límite de Tiempo



Como `Mix_PlayChannel` pero lo reproduce solo para ticks mili-segundos.

# Reproducir con efecto Fade-In

```
Mix_FadeInChannel(...);  
Mix_FadeInChannelTimed(...);
```

Son como `Mix_PlayChannel` y `Mix_PlayChannelTimed`, pero tienen efecto de “Fade In”, es decir empezando con volumen 0 y cambiando lo continuamente hasta el volumen máximo ...

# Pausar y Reanudar Un canal

```
Mix_Pause(int channel);  
Mix_Resume(int channel);
```

Pausar y reanudar un canal, es decir cuando está en pausa no entra en la mezcla, etc. El valor -1 (como parámetro) refiere a todos los canales ...

# Parar un Canal

Mix\_HaltChannel(int channel);

Parar un canal, es decir  
borra su contenido

Mix\_ExpireChannel(int channel, int ticks);

Mix\_FadeOutChannel(int channel, int ticks);

Ejecutar efecto Fade Out durante  
ticks ms y después parar el canal

Parar un canal, después  
de ticks ms

Mix\_ChannelFinished(void (\*f)(int channel));

Recibe una función como parámetro y registra esa  
función para llamarla al parar un canal (el canal se  
pasa como parámetro a esa función)

# Controlar Volumen

```
int Mix_Volume(int channel, int volume);
```

Cambiar el volumen de un canal, -1 refiere a todos los canales. El volumen puede ser entre 0 y `MIX_MAX_VOLUME`, devuelve el volumen anterior ...

```
int Mix_VolumeChunk(Mix_Chunk *chunk, int volume);
```

Fijar el volumen de un chunk, devuelve el volumen anterior ...

# Consultar información de un Canal

```
int Mix_Playing(int channel);  
int Mix_Paused(int channel);
```

Devuelven 0 o 1 si el canal está en marcha/pausa o no. En caso de pasar -1 devuelvo el numero de canales que están en marcha/pausa

```
Mix_FADING Mix_FadingChannel(int channel);
```

Devuelve el estado de "fading" de canal : MIX\_NO\_FADING, MIX\_FADE\_IN, MIX\_FADE\_OUT

```
Mix_Chunk *Mix_GetChunk(int channel);
```

Devuelve el "chunk" actual del canal

# Music

# Cargar un Archivo de Música

Carga el archivo de música (sólo una vez, se puede usar varias veces después). Devuelve null en caso de error.

```
Mix_Music *music = Mix_LoadMUS("ImperialMarch.wav");
```

```
//...  
// Usar music para reproducir el sonido, etc,  
// ...
```

```
Mix_FreeMusic(music);
```

Al final liberar la memoria correspondiente, etc.

# Reproducir Música

musica

repeticiones, -1 para reproducir infinitamente

```
Mix_PlayMusic(Mix_Music *music, int loops);
```

```
Mix_FadeInMusic(Mix_Music *music, int loops, int ticks);
```

Con efecto Fade In durante ticks ms

```
Mix_FadeInMusicPos(Mix_Music *music, int loops, int ms,  
double pos);
```

La posición de donde empezar, eso depende del formato de la música, normalmente mili-segundos desde el principio – ver documentación

# Pausar, Reanudar, etc.

```
Mix_PauseMusic();  
Mix_ResumeMusic();  
Mix_RewindMusic();  
Mix_SetMusicPosition(double pos);
```



Pausar, reanudar, rebobinar y cambiar  
la posición ....

# Controlar Volumen

```
int Mix_VolumeMusic(int volume);
```



Cambiar el volumen de la música. El volumen puede ser entre 0 y **MIX\_MAX\_VOLUME**, devuelve el volumen anterior ...

# Parar la Música

```
Mix_HaltMusic();
```

```
Mix_FadeOutMusic(int ticks);
```

Parar la musica ...

Ejecutar efecto Fade Out durante  
ticks ms

```
Mix_MusicFinished(void (*f)());
```

Recibe una función como parámetro y registra esa función para llamarla al parar la música ...

# Consultar Información

```
int PlayingMusic();  
int PausedMusic();  
int FadingMusic();
```

Como en el caso de sound effects ...

```
Mix_MusicType Mix_MusicType(Mix_Music *music);
```

Devuelve el tipo de la música:

MUS\_NONE, MUS\_CMD, MUS\_WAV, MUS\_MOD, MUS\_MID, MUS\_OGG,  
MUS\_MP3, MUS\_MP3\_MAD\_UNUSED, MUS\_FLAC,

# Más de SDL\_Mixer

- ◆ Manejo de Errores ...
- ◆ Grupos de Canales ...
- ◆ Ver la documentación ...