# PROGRAMACIÓN DE AUDIO

Máster en Programación de
Videojuegos

# TEMA 2: Efectos de sonido.

Javier Alegre Landáburu

# Contenido

Tema 2. Efectos de sonido

- Efecto doppler

- EFX
  - Efecto
  - Slot
  - Filtro

# Efecto doppler

- El efecto doppler es un fenómeno físico producido por la distorsión de las ondas como consecuencia del movimiento de cuerpos.

- Este efecto se puede visualizar al lanzar un objeto al agua. Se crean ondas que se alejan de forma radial del origen del impacto.

# Efecto doppler

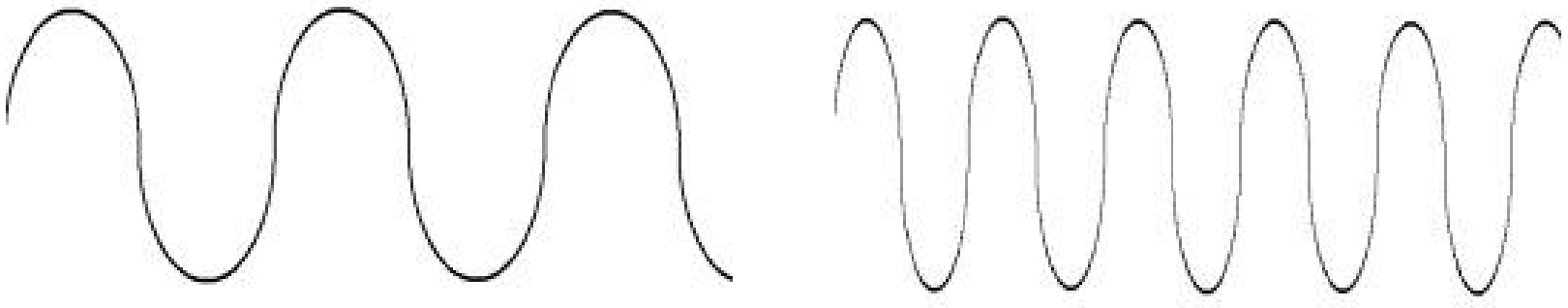
# Efecto doppler

- Las ondas sonoras producen el mismo efecto. Cuando un objeto emite sonido, las ondas se dispersan desde la fuente de sonido.

- Si el objeto que emite sonido, o el objeto que lo percibe (el oyente) están en movimiento, se produce una distorsión en la frecuencia de la onda (shifting).

# Efecto doppler

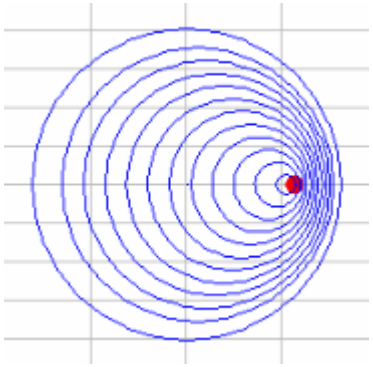
- La frecuencia de la onda aumenta en la dirección en la que el emisor de sonido se desplaza.

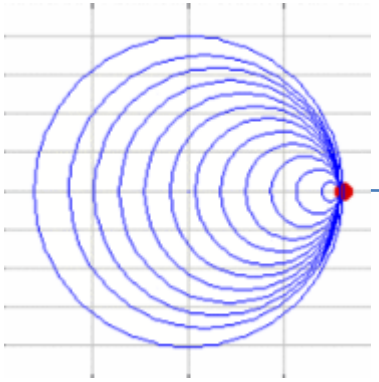


# Efecto doppler

- De igual forma, la frecuencia de la onda disminuye en la dirección opuesta al movimiento del emisor.

- Este fenómeno afecta a la forma en la que el oyente percibe el sonido.





Sonic Boom

Mach 1

# Efecto doppler

- OpenAL simula el efecto doppler, utilizando el algoritmo siguiente:

```
shift = DOPPLER_FACTOR * freq * (DOPPLER_VELOCITY -
listener.velocity) / (DOPPLER_VELOCITY - source.velocity)
```

- Estos parámetros se establecen con las siguientes funciones:

# Efecto doppler

```
void alDopplerFactor(ALfloat factor)
```

Establece el valor de DOPPLER_FACTOR. El parámetro debe ser positivo. Modificamos la magnitud de la ecuación de la siguiente forma:

- **factor = 0**: Desactiva el efecto doppler.
- **0 < factor < 1**: Disminuye el efecto doppler.
- **factor = 1**: Valor del efecto doppler por defecto.
- **factor > 1**: Intensifica el efecto doppler.

# Efecto doppler

```
void alDopplerVelocity(ALfloat velocity)
```

Establece el valor de DOPPLER_VELOCITY, que indica la velocidad a la que se desplaza el sonido (cambia el medio en el que el sonido se desplaza -aire, agua…-).

El valor por defecto es 343.3 m/s, que corresponde a la velocidad del sonido en el aire (20º C, al 50% de humedad y a nivel del mar).

# EFX

- Las tarjetas de audio soportaban la simulación de ciertos efectos sonoros mediante hardware. Las extensiones para soportar estos efectos se llaman Environmental Audio Extensions (EAX).

- Desde la llegada de Windows Vista, se desactivó el soporte para EAX por hardware, y ha caído en desuso.

# EFX

- No obstante, OpenAL Soft soporta un conjunto de extensiones para implementar estos efectos mediante software. A estas extensiones se les llama Effects Extension (EFX).

- ¿Qué tipo de efectos se pueden conseguir con EFX?

# EFX

- Imaginemos un juego donde el personaje se mueve de espacios abiertos a espacios cerrados, se sumerge en el agua, etc.

- Si un jugador lanza una granada dentro de una casa, el sonido debe oírse de diferente manera para el que está en el interior que para el que lo oye desde fuera.

# EFX

- De igual forma, el sonido no se percibe de la misma manera estando bajo el agua que en la superficie.
- Con OpenAL, podremos definir efectos de reverberación, distorsión, etc.
- Veamos los elementos de OpenAL necesarios para producir estos efectos:

# Efecto

- Un efecto (effect) modifica el sonido con un tipo de efecto y unos parámetros determinados. Por ejemplo, podríamos crear un efecto de reverberación, y según sus parámetros, simular los entornos de un baño, una cueva, agua…

# Efecto

- Un efecto se crea con la siguiente función:

```
void alGenEffects(ALsizei n, ALuint* effects)
```

- Y se destruye con:

```
void alDeleteEffects(ALsizei n, ALuint* effects)
```

# Efecto

- El tipo de efecto a simular se establece con la función:

```
void alEffecti(ALuint effect, ALenum param, ALint value)
```

- El valor de param debe ser `AL_EFFECT_TYPE`.

- Se pueden simular muchos tipos de efectos: `AL_EFFECT_REVERB`, `AL_EFFECT_CHORUS`, `AL_EFFECT_DISTORTION`, `AL_EFFECT_ECHO`, `AL_EFFECT_FLANGER`…

# Slot

- Un slot es un contenedor para un efecto. Se puede ligar una fuente de sonido a un slot determinado, y el sonido emitido por esa fuente será afectado por el efecto del slot.

# Slot

- Crearemos slots con la función:

```
void alGenAuxiliaryEffectSlots(ALsizei n, ALuint* slots)
```

- Y los destruiremos con:

```
void alDeleteAuxiliaryEffectSlots(ALsizei n, ALuint* slots)
```

# Slot

- Para fijar un efecto en el slot, haremos:

```
alAuxiliaryEffectSloti(ALuint slot, ALenum param, ALint value)
```

- Pasaremos AL_EFFECTSLOT_EFFECT como valor de *param*, y el efecto correspondiente como *value*.

# Slot

- Es necesario ligar una fuente al slot cuyo efecto queremos aplicar. Esto se hace con la función:

```
void alSource3i(ALuint source, ALenum param, ALint v1,
ALint v2, ALint v3)
```

- El valor de *param* debe ser AL_AUXILIARY_SEND_FILTER, *v1* será el slot a ligar, y *v2* y *v3* serán 0.

# Filtro

- Un filtro permite pasar un efecto establecido por una serie de modificadores, que permiten simular por ejemplo que el sonido lo estamos escuchando a través de una pared, o por medio de un teléfono.

# Filtro

- Creamos un filtro con la función:

```
void alGenFilters(ALsizei n, ALuint* filters);
```

- Y lo destruiremos con:

```
void alDeleteFilters(ALsizei n, ALuint* filters);
```

# Filtro

- Una vez creado el filtro, indicaremos el tipo de filtrado que debe de realizar, utilizando la función:

```
void alFilteri(ALuint filter, ALenum param, ALint value)
```

- El valor de *param* debe ser `AL_FILTER_TYPE`, y *value* puede ser `AL_FILTER_NULL`, `AL_FILTER_LOWPASS`, `AL_FILTER_HIGHPASS`, `AL_FILTER_BANDPASS`.

# Filtro

- Un filtro puede modificar la onda de un emisor directamente, o la onda que es enviada a un slot.

- Para hacer lo primero, utilizaremos `alSourcei` con el parámetro `AL_DIRECT_FILTER`.

- Para hacer lo segundo, cuando establecemos el slot con `alSource3i`, los dos últimos parámetros deberán ser 1 y el identificador del filtro, respectivamente.

# Ejemplo

- Ejemplo de uso de Effects Extension:

Hay que incluir “efx.h”.

Para poder usar las funciones tenemos que asignar punteros a función, por ejemplo para geneffects:

```
LPALGENEFFECTS  alGenEffects  = nullptr;

alGenEffects   =
(LPALGENEFFECTS)alGetProcAddress("alGenEffects");
```

# Ejemplo

- Hay que mirar en efx.h:

```
/* Effect object function types. */
typedef void (AL_APIENTRY *LPALGENEFFECTS)(ALsizei, ALuint*);
typedef void (AL_APIENTRY *LPALDELETEEFFECTS)(ALsizei, const ALuint*);
typedef ALboolean (AL_APIENTRY *LPALISEFFECT)(ALuint);
typedef void (AL_APIENTRY *LPALEFFECTI)(ALuint, ALenum, ALint);
typedef void (AL_APIENTRY *LPALEFFECTIV)(ALuint, ALenum, const ALint*);
typedef void (AL_APIENTRY *LPALEFFECTF)(ALuint, ALenum, ALfloat);
typedef void (AL_APIENTRY *LPALEFFECTFV)(ALuint, ALenum, const ALfloat*);
typedef void (AL_APIENTRY *LPALGETEFFECTI)(ALuint, ALenum, ALint*);
typedef void (AL_APIENTRY *LPALGETEFFECTIV)(ALuint, ALenum, ALint*);
typedef void (AL_APIENTRY *LPALGETEFFECTF)(ALuint, ALenum, ALfloat*);
typedef void (AL_APIENTRY *LP
```

```
#ifdef AL_ALEXT_PROTOTYPES
AL_API ALvoid AL_APIENTRY alGenEffects(ALsizei n, ALuint *effects);
AL_API ALvoid AL_APIENTRY alDeleteEffects(ALsizei n, const ALuint *effects);
AL_API ALboolean AL_APIENTRY alIsEffect(ALuint effect);
AL_API ALvoid AL_APIENTRY alEffecti(ALuint effect, ALenum param, ALint iValue);
AL_API ALvoid AL_APIENTRY alEffectiv(ALuint effect, ALenum param, const ALint *piValues);
AL_API ALvoid AL_APIENTRY alEffectf(ALuint effect, ALenum param, ALfloat flValue);
AL_API ALvoid AL_APIENTRY alEffectfv(ALuint effect, ALenum param, const ALfloat *pflValues);
AL_API ALvoid AL_APIENTRY alGetEffecti(ALuint effect, ALenum param, ALint *piValue);
AL_API ALvoid AL_APIENTRY alGetEffectiv(ALuint effect, ALenum param, ALint *piValues);
AL_API ALvoid AL_APIENTRY alGetEffectf(ALuint effect, ALenum param, ALfloat *pflValue);
AL_API ALvoid AL_APIENTRY alGetEffectfv(ALuint effect, ALenum param, ALfloat *pflValues);
```