3D Game Programming
Audio

Ming-Te Chi
Department of Computer Science,
National Chengchi University

Goal

- Sound and music in game
- Unity3D Audio
  – Listener, source, clip
  – Play, stop, pause
  – effect

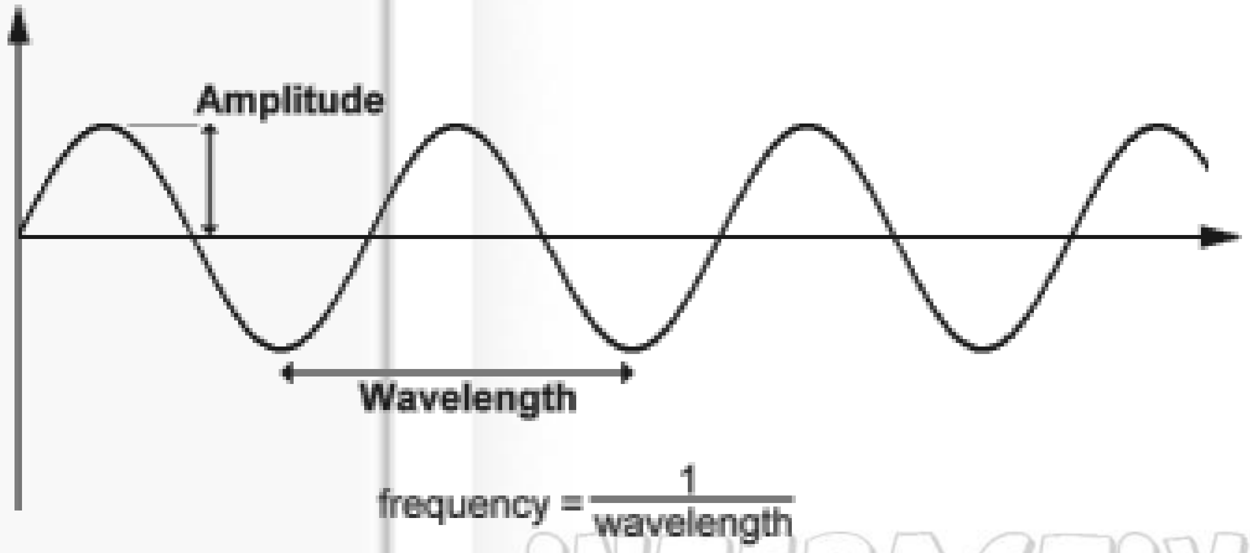
- OpenAL programming

Music and Sound

- Game music
  - Transition animation
- Sound FX
  - Exploding
  - Wind blowing
  - Raining
  - Walking

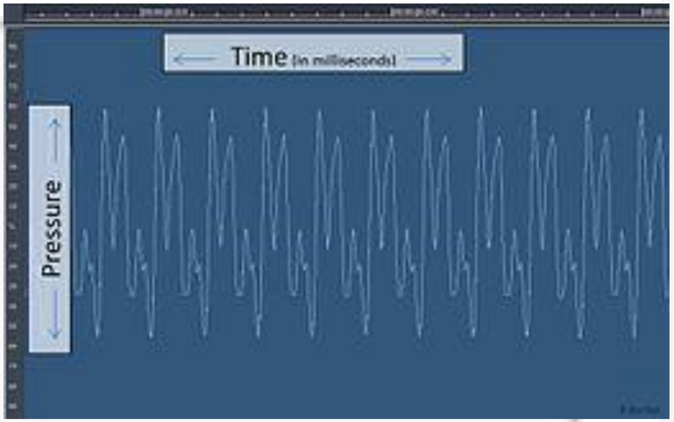
Sound wave

- Sound waves are often simplified to a description in terms of sinusoidal plane waves



HEY, LOOK!
I CAN SEE MY VOICE!
* LAUGHS *

Sound



Properties of sound wave

- Frequency, or its inverse, the period
  - Wavelength
  - Wavenumber
- Amplitude
- Sound pressure
- Sound intensity
- Speed of sound source
- Direction

Audio Format

- An audio format is a medium for storing sound and music.
- Wav and mp3 are common format
- Store 44,100 samples per second, 16 bits per sample.

Windows api

```
BOOL WINAPI
PlaySound( LPCSTR pszSound,
           HMODULE hmod,
           DWORD fdwSound );
```

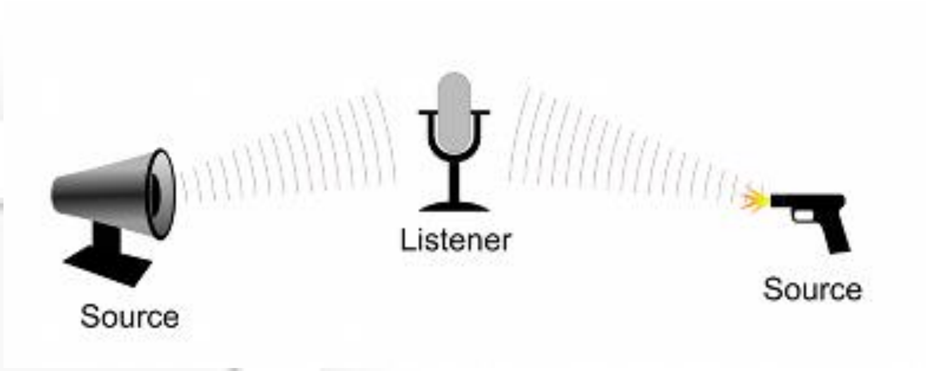
- Simply use windows api to play wav

- Ex: play a wav file in “music/ding.wav”

```
PlaySound("music/ding.wav",
NULL,SND_ASYNC | SND_FILENAME);
```

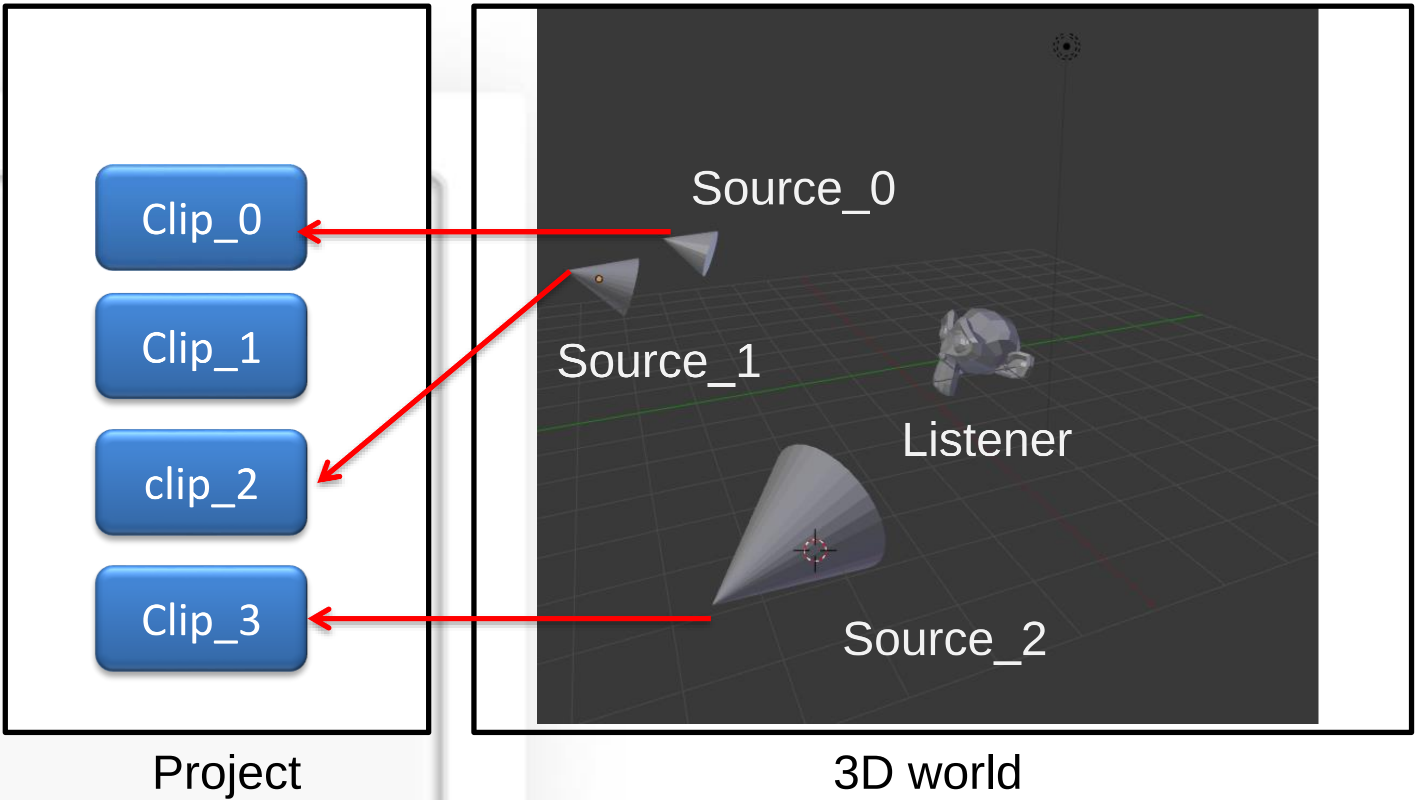
AUDIO IN UNITY3D

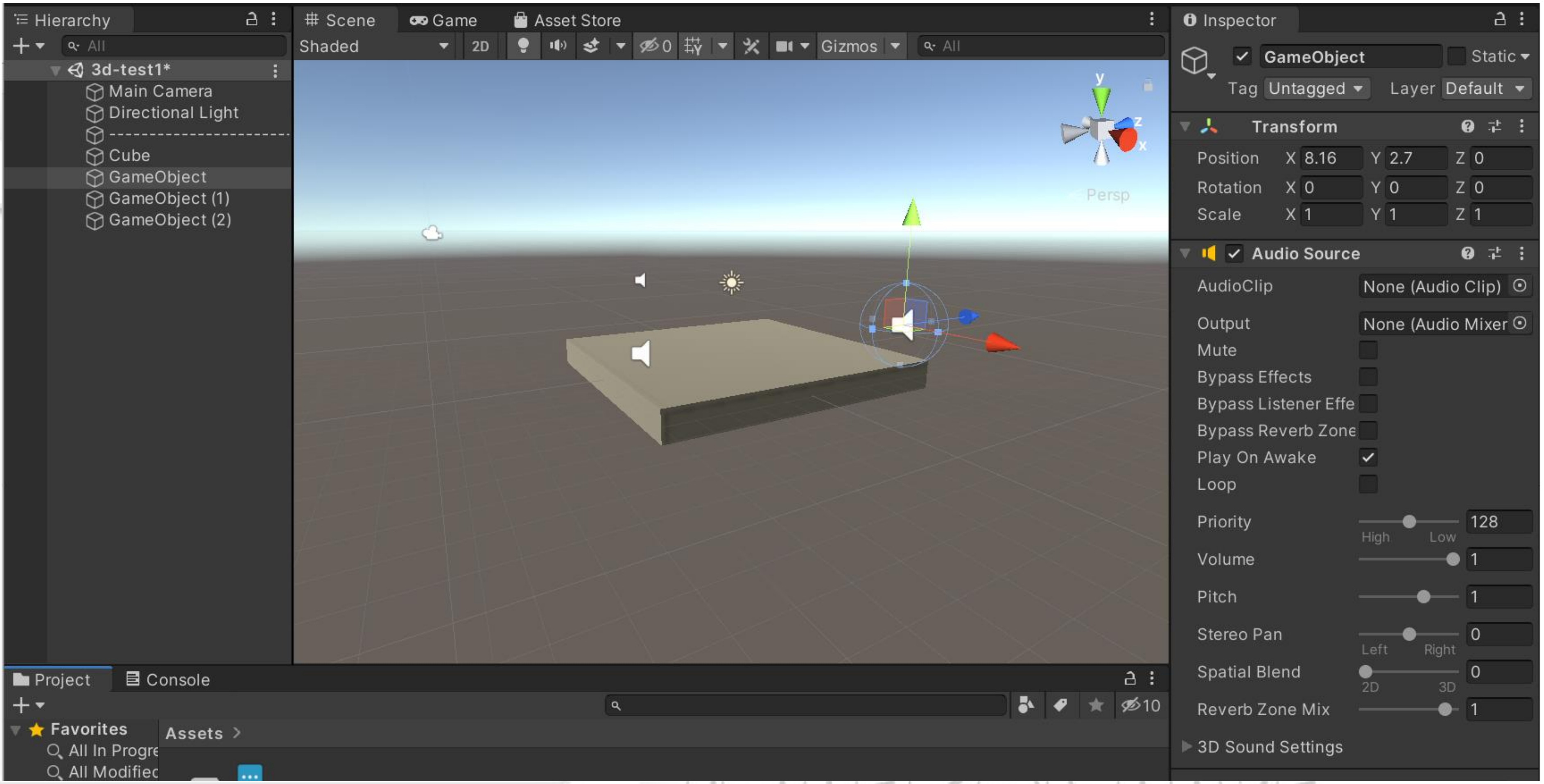
Audio in Unity3D



- Audio Sources attached to objects,
- Audio Listener attached to another object, most often the main camera

Architecture





GameObject Static
Tag Untagged Layer Default

Transform

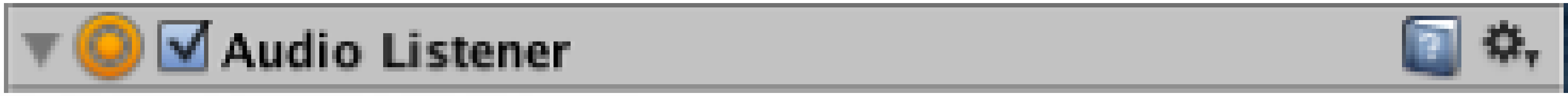
Position	X 8.16	Y 2.7	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Audio Source

AudioClip	None (Audio Clip)
Output	None (Audio Mixer
Mute	
Bypass Effects	
Bypass Listener Effe	
Bypass Reverb Zone	
Play On Awake	✓
Loop	
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

3D Sound Settings

Audio Listener

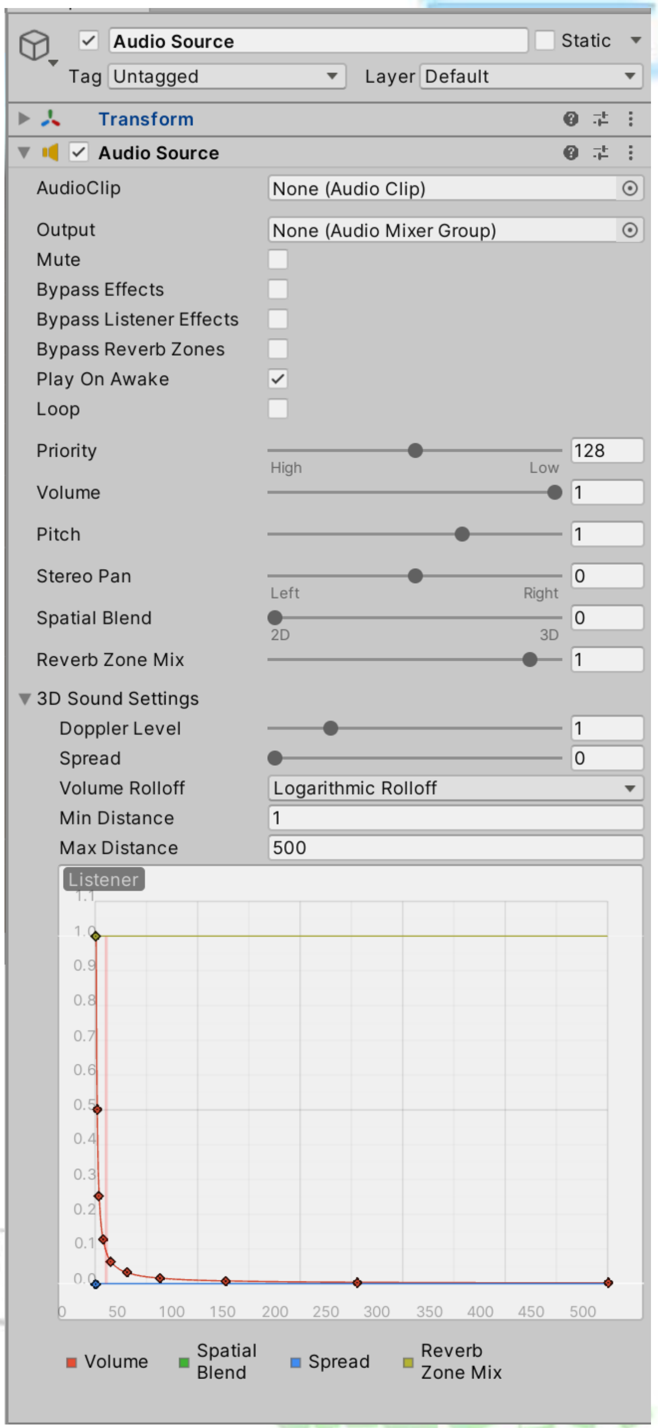


Each scene can only have 1 Audio Listener to work properly.

Audio Source

- Plays back an [Audio Clip](#) in the scene.

- Volume
- Pitch



Audio Source ✓ | Static
Tag Untagged | Layer Default

Transform

✓ Audio Source

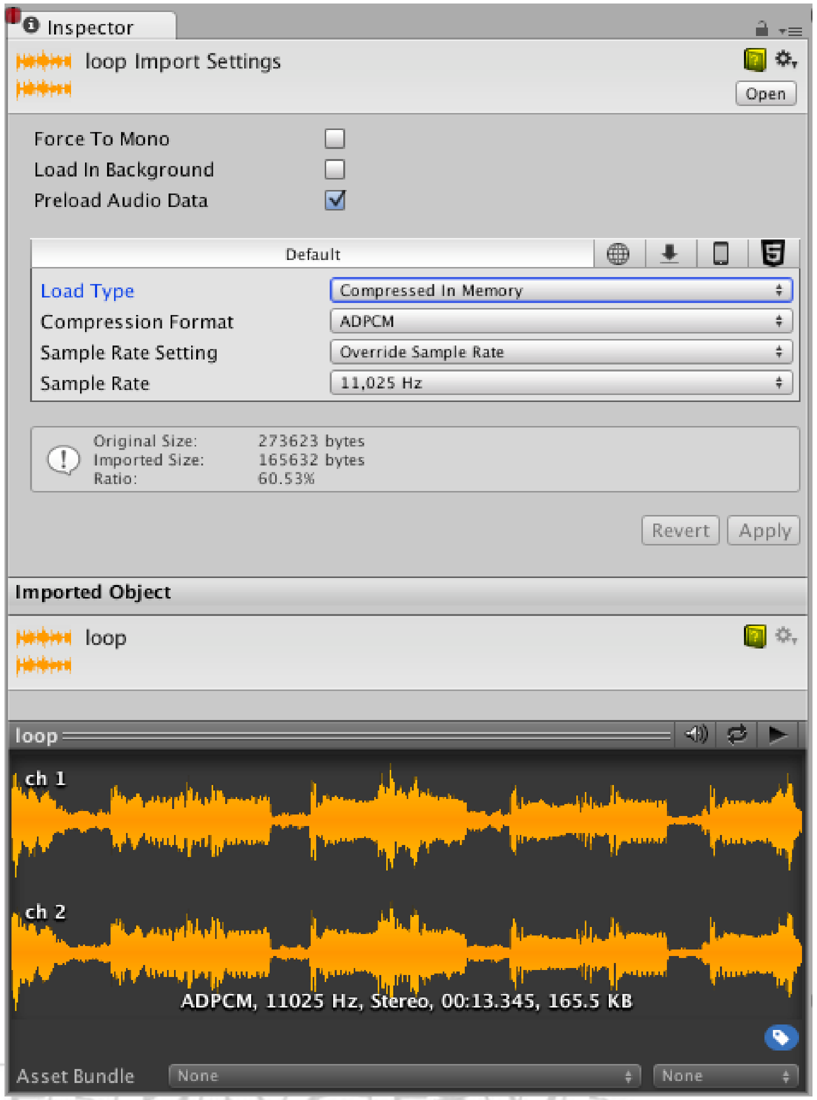
AudioClip	None (Audio Clip)
Output	None (Audio Mixer Group)
Mute	
Bypass Effects	
Bypass Listener Effects	
Bypass Reverb Zones	
Play On Awake	✓
Loop	
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

3D Sound Settings

Doppler Level	1
Spread	0
Volume Rolloff	Logarithmic Rolloff
Min Distance	1
Max Distance	500

Audio Clip

- In Project view


Inspector

loop Import Settings

Open

Force To Mono ☐
Load In Background ☐
Preload Audio Data ✓

Default

Load Type: Compressed In Memory
Compression Format: ADPCM
Sample Rate Setting: Override Sample Rate
Sample Rate: 11,025 Hz

Original Size: 273623 bytes
Imported Size: 165632 bytes
Ratio: 60.53%

Revert Apply

Imported Object

loop

ADPCM, 11025 Hz, Stereo, 00:13.345, 165.5 KB

Asset Bundle: None None

- Support file format:
  - AIFF, WAV, MP3 and Ogg formats

Play()/Stop()

```
void Start() {
        AudioSource audio = GetComponent<AudioSource>();
        audio.Play();
        audio.Play(44100);
   }
```

Pause()/UnPause()

Play, wait, switch clip

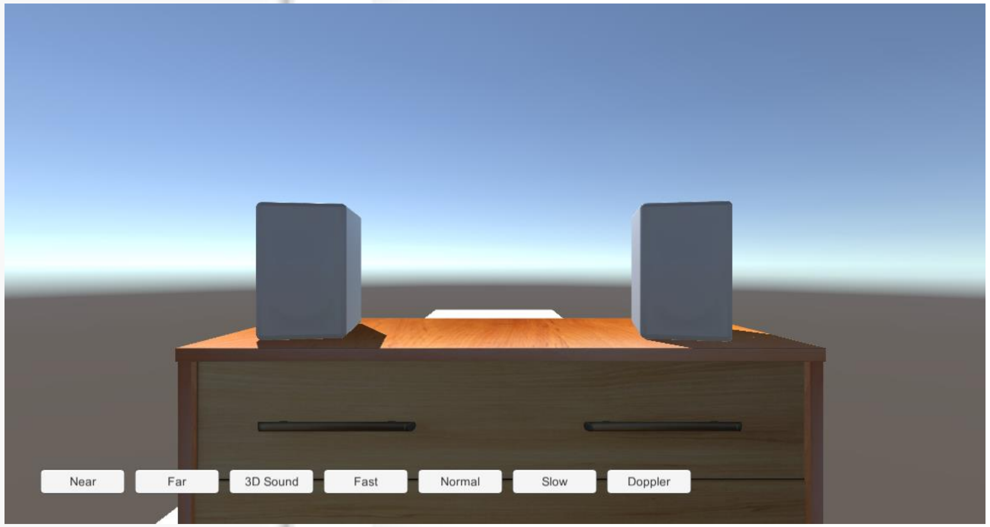
```
[RequireComponent(typeof(AudioSource))]
public class ExampleClass : MonoBehaviour {
    public AudioClip otherClip;

    IEnumerator Start() {
        AudioSource audio = GetComponent<AudioSource>();

        audio.Play();
        yield return new WaitForSeconds(audio.clip.length);
        audio.clip = otherClip;
        audio.Play();
    }
}
```

AudioSource.unity

- 嘗試各種不同的AudioSource效果。



AudioSource

- 有時候，某些事件發生後會需要播放一段短暫的音效。
  - UI按鈕音效。
  - 爆炸聲。
  - 角色被攻擊的慘叫。
- AudioSource.playOneShot(適合參數一樣的音效，如UI音效，或主角發出的所有聲音)
- AudioSource.playClipAtPoint(適合即時產生在場景某處的3D音效，如爆炸聲)

AudioSource.PlayOneShot

- 需要一個AudioSource Component。
- 使用該Audio Source，播放一段Audio Clip。
- 不會更改該Audio Source目前的clip。
- Audio Source任何的參數變化/狀態變化都會影響播放中的OneShot。
- 可以同時呼叫PlayOneShot多次，彼此之間不互相影響。
- 播放以後沒辦法獨立暫停/取消。

AudioSource.PlayClipAtPoint

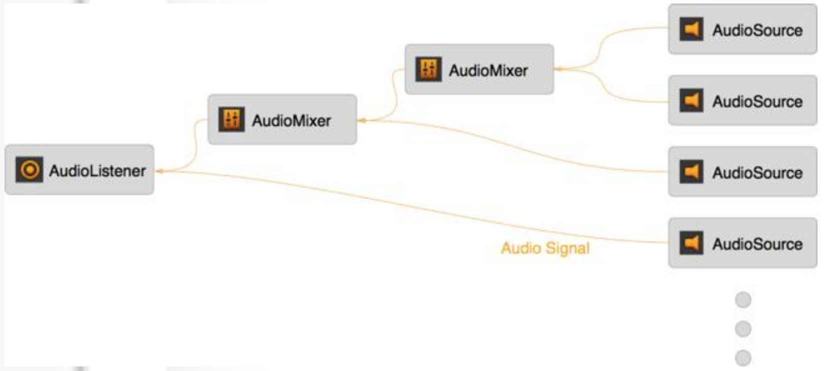
- AudioSource class的靜態成員函式，不需要AudioSource實體。
- 在世界座標指定的位址播放一段Audio Clip。必定為3D Sound。
- 場景中會產生一個One shot audio物件，播放完會自我刪除。

Audio Effects

- 本身不發出聲音，但可以影響Audio Source發出的聲音或Audio Listener聽到的聲音。
- 音效元件:
  - Audio Mixer + Audio Effect
  - Audio Filter
  - Reverb Zone

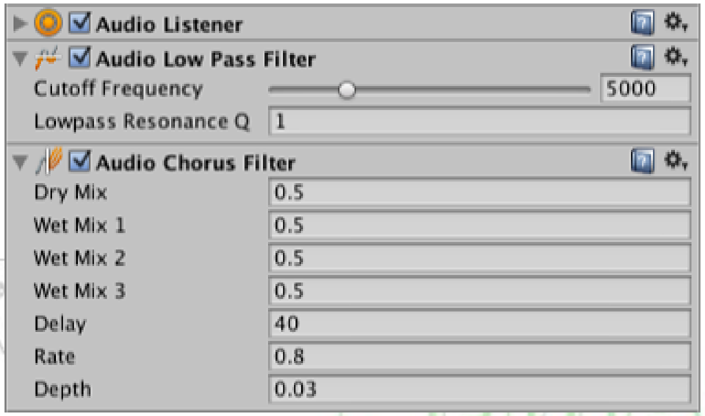
Audio Mixer + Audio Effect

- 控制Audio Source混合的順序，每一次混和都可以加入不同的Audio Effect。
- 請參閱文件: http://docs.unity3d.com/Manual/AudioMixer.html



Audio Filter

- 直接對物件發出/聽到的音效做出filter效果。
- 對Audio Source使用：影響發出的聲音。
- 對Audio Listener使用：影響聽到的所有聲音。
- Component的順序會影響filter套用的順序。
- 圖：對Audio Listener聽到的所有聲音，先做low pass(過濾掉頻率高於5000hz的聲音)，再做chorus(合唱) filter。



Audio Listener

Audio Low Pass Filter

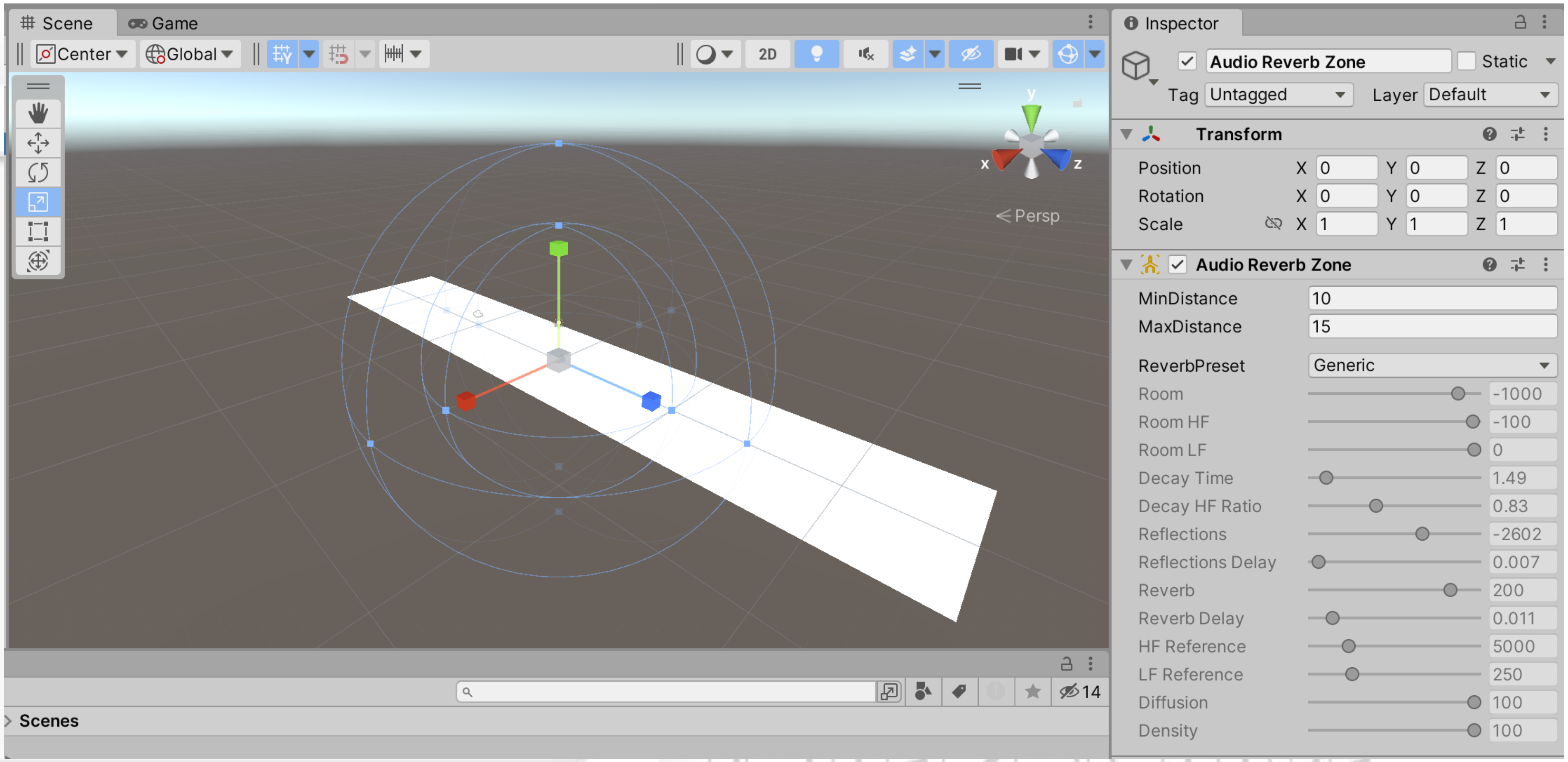
Cutoff Frequency	5000
Lowpass Resonance Q	1

Audio Chorus Filter

Dry Mix	0.5
Wet Mix 1	0.5
Wet Mix 2	0.5
Wet Mix 3	0.5
Delay	40
Rate	0.8
Depth	0.03

Reverb Zone

- Reverb，殘響，用來模擬聲音在具有障礙物的環境中的反射/折射/繞射綜合產生的複雜效果。
- 使用時機如：玩家走進山洞中時，腳步聲會產生回音。此時只要將山洞整個包在Reverb Zone中，所有位於其中的Audio Source和Audio Listener都會自動受到影響。
- 有各項環境參數可調整，與一般電腦中的混音軟體相似。



Inspector

Audio Reverb Zone ✓ Static

Tag Untagged Layer Default

Transform

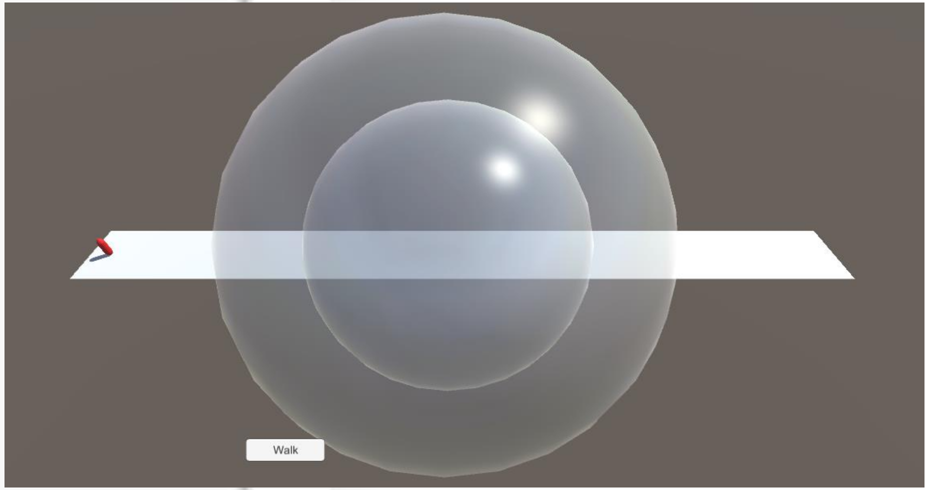
	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Audio Reverb Zone ✓

MinDistance	10
MaxDistance	15
ReverbPreset	Generic
Room	-1000
Room HF	-100
Room LF	0
Decay Time	1.49
Decay HF Ratio	0.83
Reflections	-2602
Reflections Delay	0.007
Reverb	200
Reverb Delay	0.011
HF Reference	5000
LF Reference	250
Diffusion	100
Density	100

ReverbZone.unity

- 當角色進入Reverb Zone時，腳步聲會變化，模擬洞穴的回音。



參考資料

OpenAL

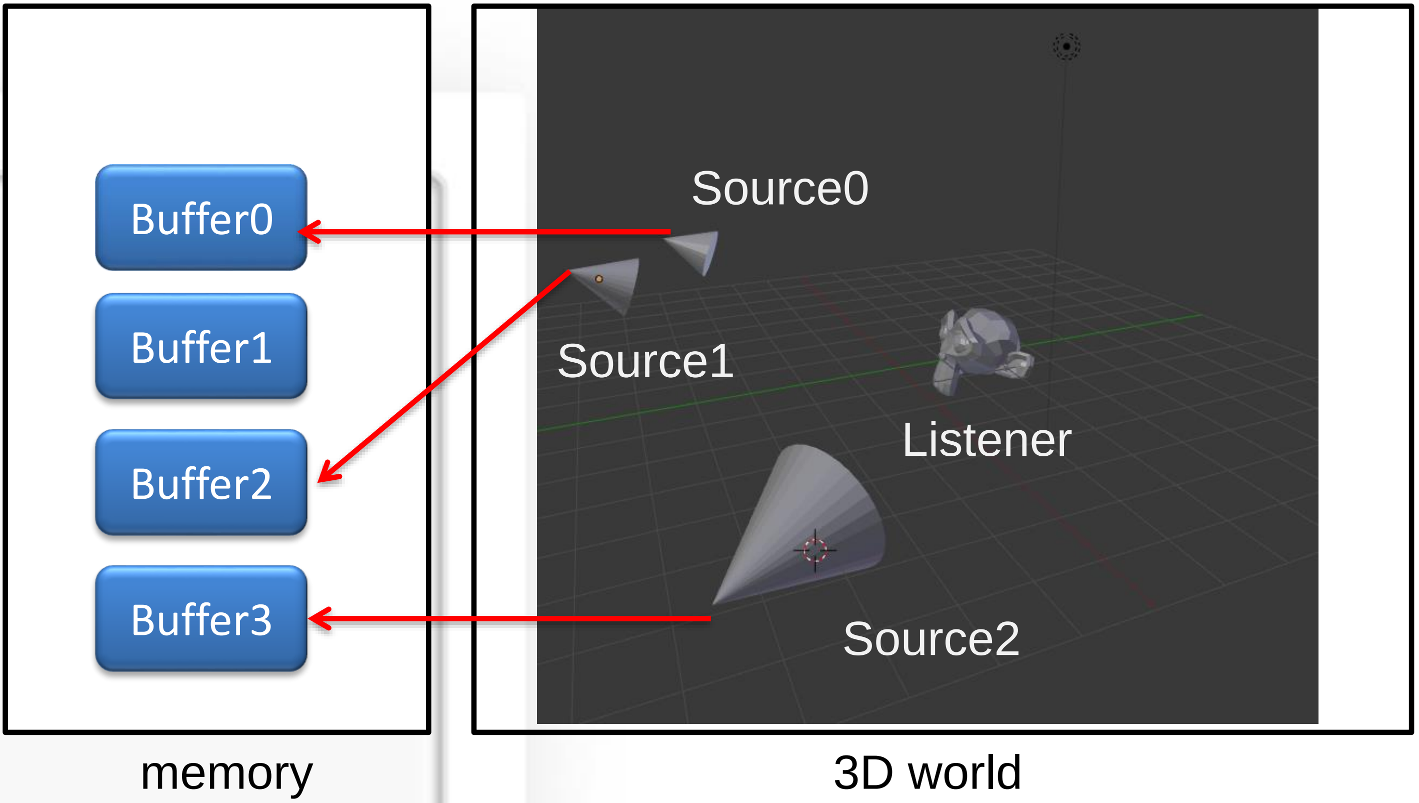
- OpenAL (**Open** **A**udio **L**ibrary) is a free software cross-platform audio API.

- It is designed for efficient rendering of multichannel three dimensional positional audio.

OpenAL Basic Elements

- **Source**
  - – A source of a sound in the world
  - – Link to buffer for actual data to play
- **Buffer**
  - – Hold physical sound data in the memory
  - – Cannot play buffer directly
    - Need to use source
- **Listener**
  - – The ears of the world

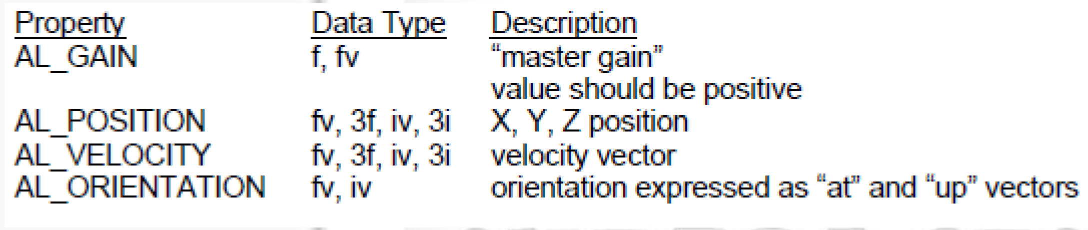
OpenAL Architecture



Listener

- For every context, there is automatically one Listener object.

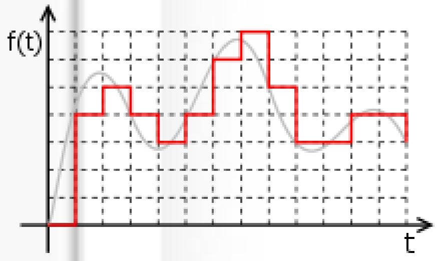
alListenerfv(AL_POSITION, listenerPos);
alListenerfv(AL_VELOCITY, listenerVel);
alListenerfv(AL_ORIENTATION, listenerOri);

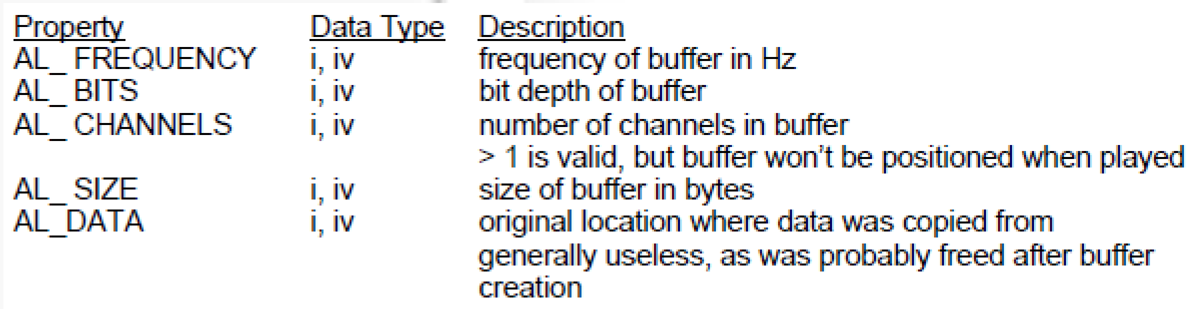


Property	Data Type	Description
AL_GAIN	f, fv	"master gain" value should be positive
AL_POSITION	fv, 3f, iv, 3i	X, Y, Z position
AL_VELOCITY	fv, 3f, iv, 3i	velocity vector
AL_ORIENTATION	fv, iv	orientation expressed as "at" and "up" vectors

Buffer

- Each buffer generated by alGenBuffers has properties which can be retrieved.





Property	Data Type	Description
AL_FREQUENCY	i, iv	frequency of buffer in Hz
AL_BITS	i, iv	bit depth of buffer
AL_CHANNELS	i, iv	number of channels in buffer > 1 is valid, but buffer won't be positioned when played
AL_SIZE	i, iv	size of buffer in bytes
AL_DATA	i, iv	original location where data was copied from generally useless, as was probably freed after buffer creation

```
const int NUM_BUFFERS = 3;
ALuint    buffer[NUM_BUFFERS];

ALboolean al_bool;
ALsizei size,freq;
ALenum format;
ALvoid *data = NULL;
int ch;
```

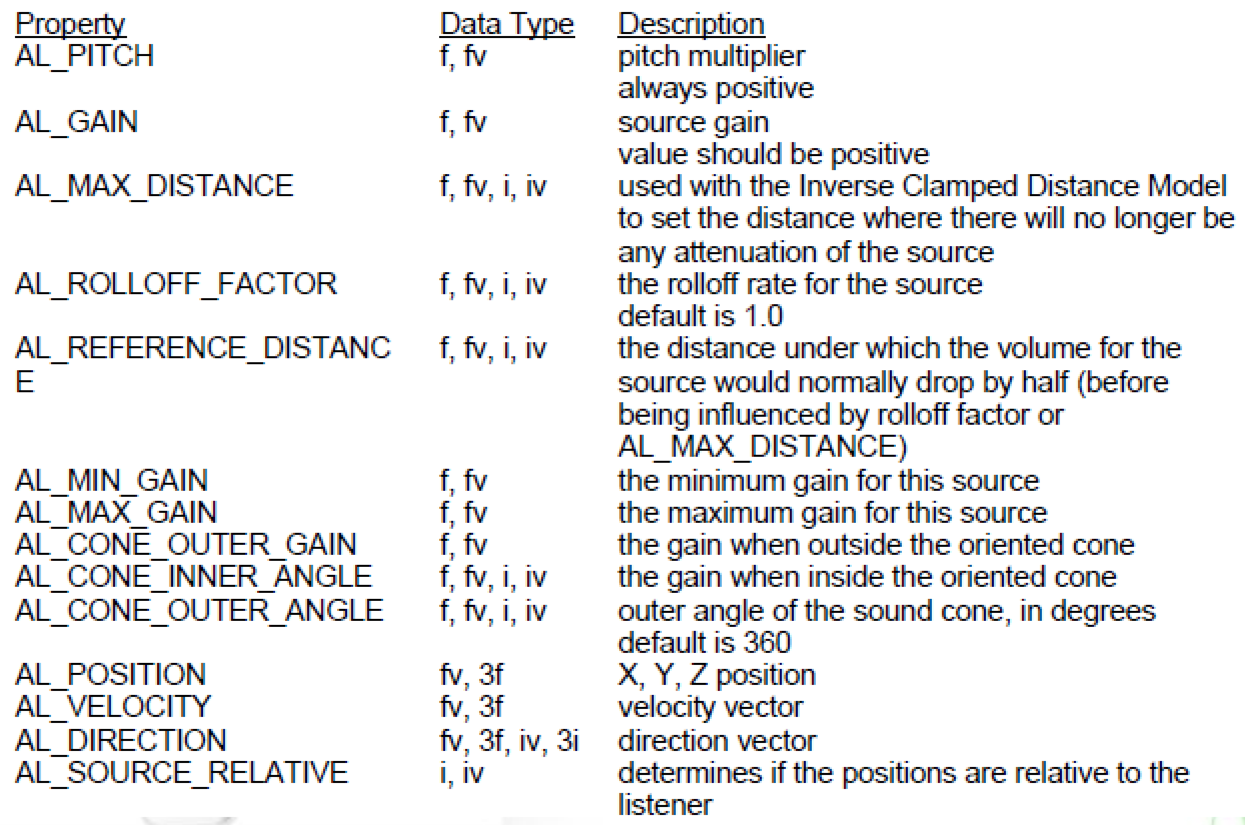
```
    // Generate buffers, or no sound will be produced
    alGenBuffers(NUM_BUFFERS, buffer);

    if(alGetError() != AL_NO_ERROR) {
            printf("- Error creating buffers !!\n");
            exit(1);
    } else {
            // printf("Created buffers\n");
    }

    alutLoadWAVFile("c.wav", &format ,&data, &size, &freq, &al_bool);
    alBufferData(buffer[0], format, data, size, freq);
    alutUnloadWAV(format, data, size, freq);
```

Source

- A source in OpenAL is exactly what it sounds like, a source of a sound in the world.



Property	Data Type	Description
AL_PITCH	f, fv	pitch multiplier always positive
AL_GAIN	f, fv	source gain value should be positive
AL_MAX_DISTANCE	f, fv, i, iv	used with the Inverse Clamped Distance Model to set the distance where there will no longer be any attenuation of the source
AL_ROLLOFF_FACTOR	f, fv, i, iv	the rolloff rate for the source default is 1.0
AL_REFERENCE_DISTANCE	f, fv, i, iv	the distance under which the volume for the source would normally drop by half (before being influenced by rolloff factor or AL_MAX_DISTANCE)
AL_MIN_GAIN	f, fv	the minimum gain for this source
AL_MAX_GAIN	f, fv	the maximum gain for this source
AL_CONE_OUTER_GAIN	f, fv	the gain when outside the oriented cone
AL_CONE_INNER_ANGLE	f, fv, i, iv	the gain when inside the oriented cone
AL_CONE_OUTER_ANGLE	f, fv, i, iv	outer angle of the sound cone, in degrees default is 360
AL_POSITION	fv, 3f	X, Y, Z position
AL_VELOCITY	fv, 3f	velocity vector
AL_DIRECTION	fv, 3f, iv, 3i	direction vector
AL_SOURCE_RELATIVE	i, iv	determines if the positions are relative to the listener

```
const int NUM_SOURCES = 3;
ALuint    source[NUM_SOURCES];
```

```
    alGetError(); /* clear error */
    alGenSources(NUM_SOURCES, source);

    if(alGetError() != AL_NO_ERROR) {
            printf("- Error creating sources !!\n");
            exit(2);
    }

    alSourcef(source[0], AL_PITCH, 1.0f);
    alSourcef(source[0], AL_GAIN, 1.0f);
    alSourcefv(source[0], AL_POSITION, source0Pos);
    alSourcefv(source[0], AL_VELOCITY, source0Vel);
    alSourcei(source[0], AL_BUFFER, buffer[0]); //attach buffer
    alSourcei(source[0], AL_LOOPING, AL_TRUE);
```

Play and Stop

Combine with keyboardfunc(), or some other way

Ex:

- alSourcePlay(source[0]);
- alSourceStop(source[0]);
- alSourcePause(source[0]);