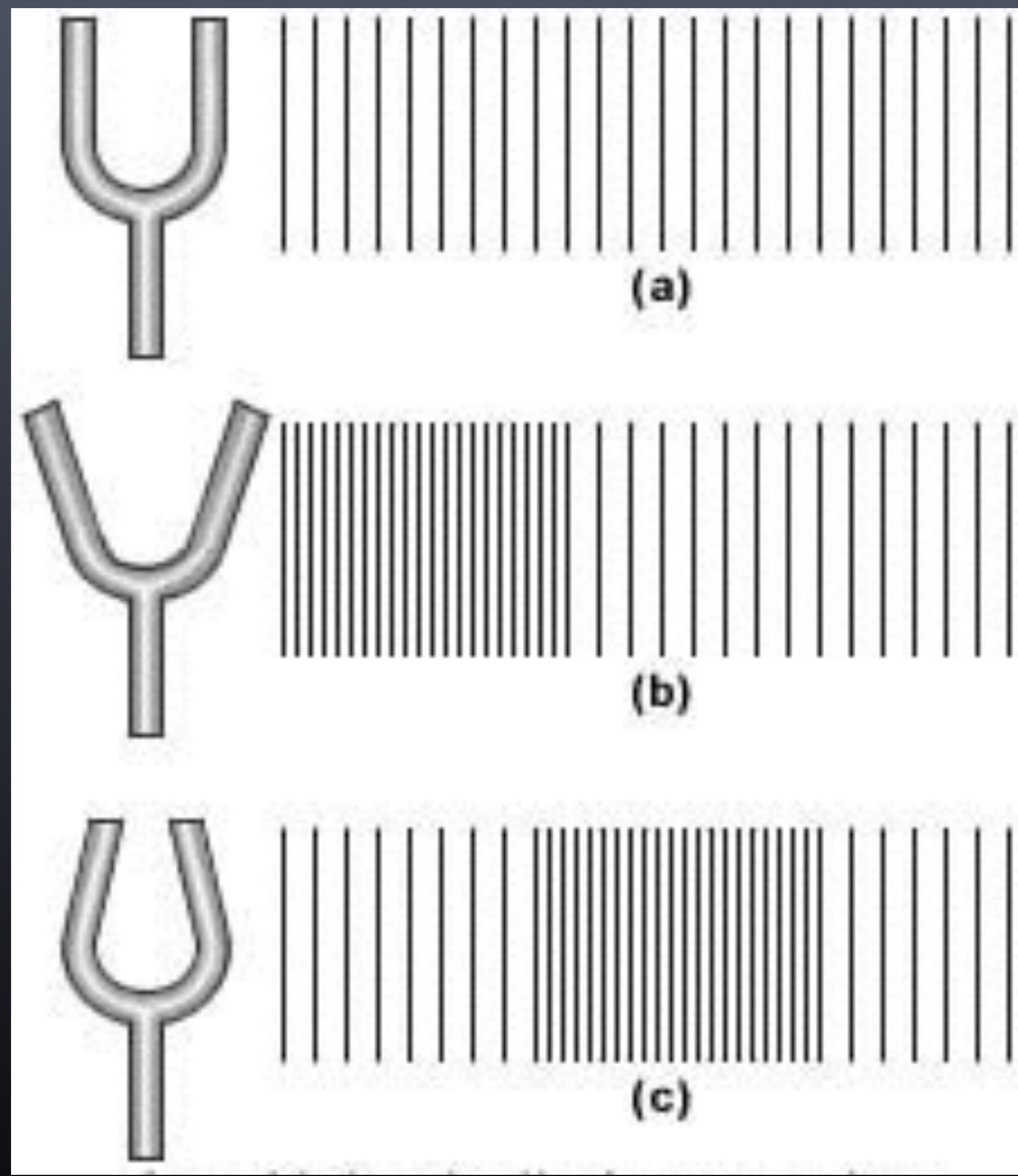
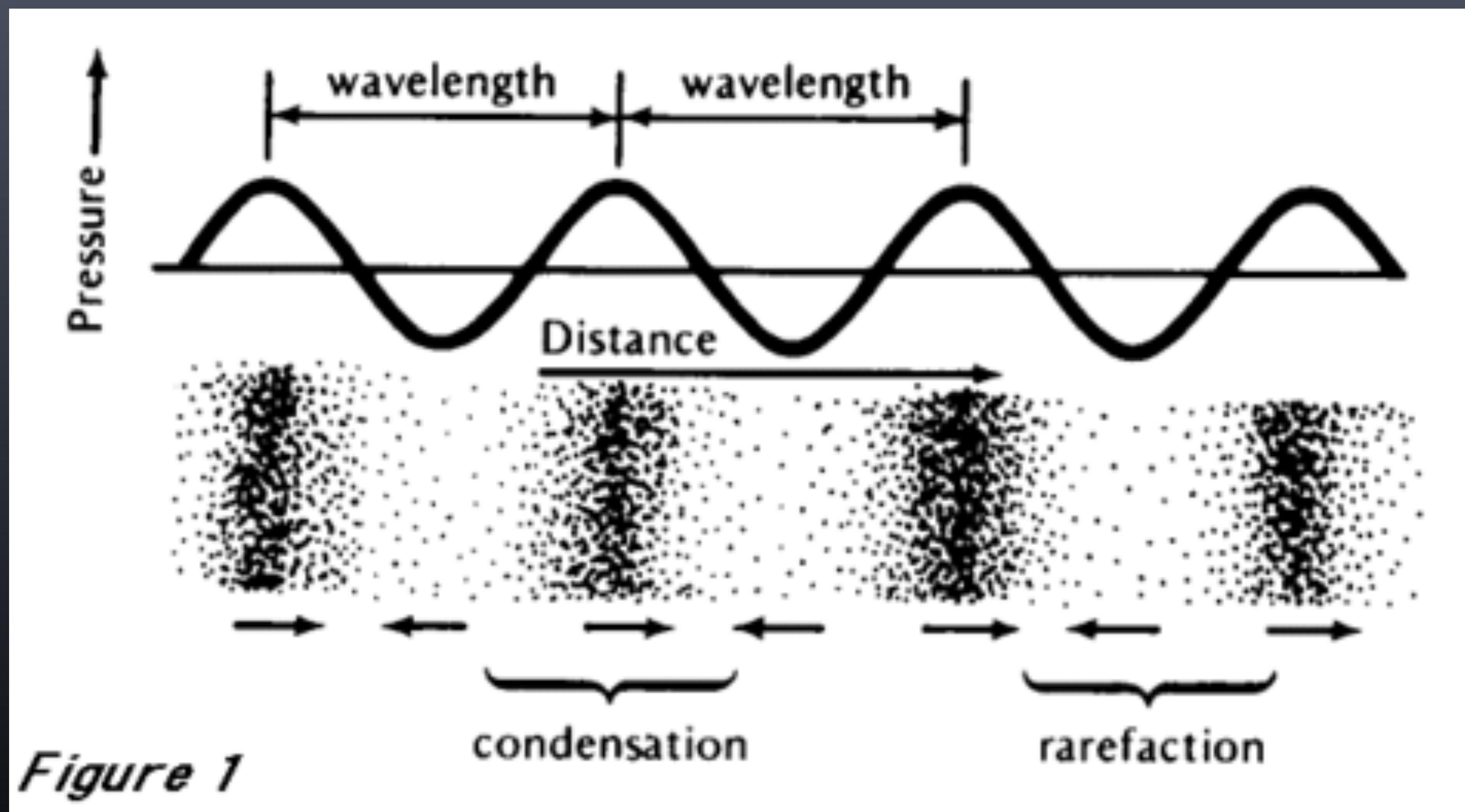


# Playing sound.

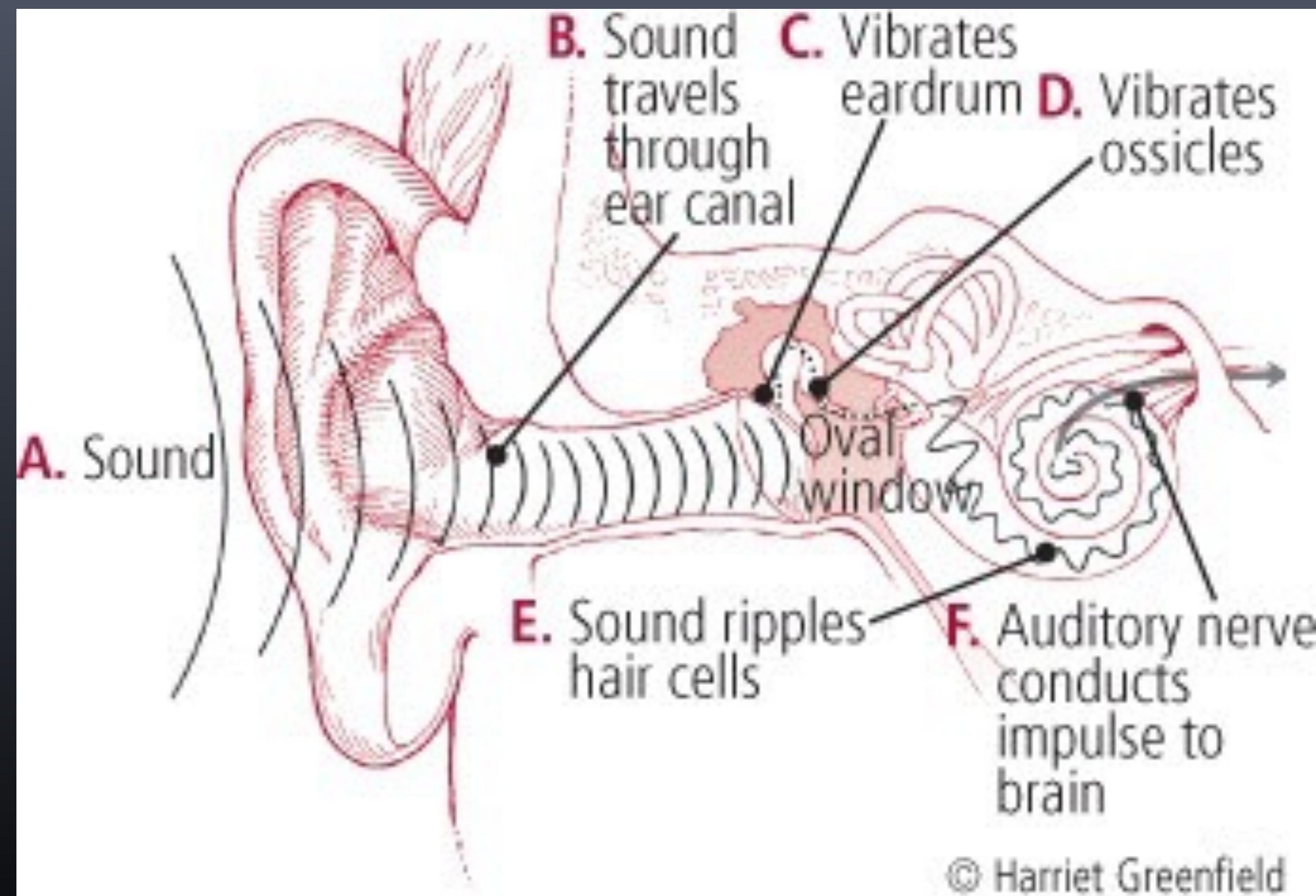


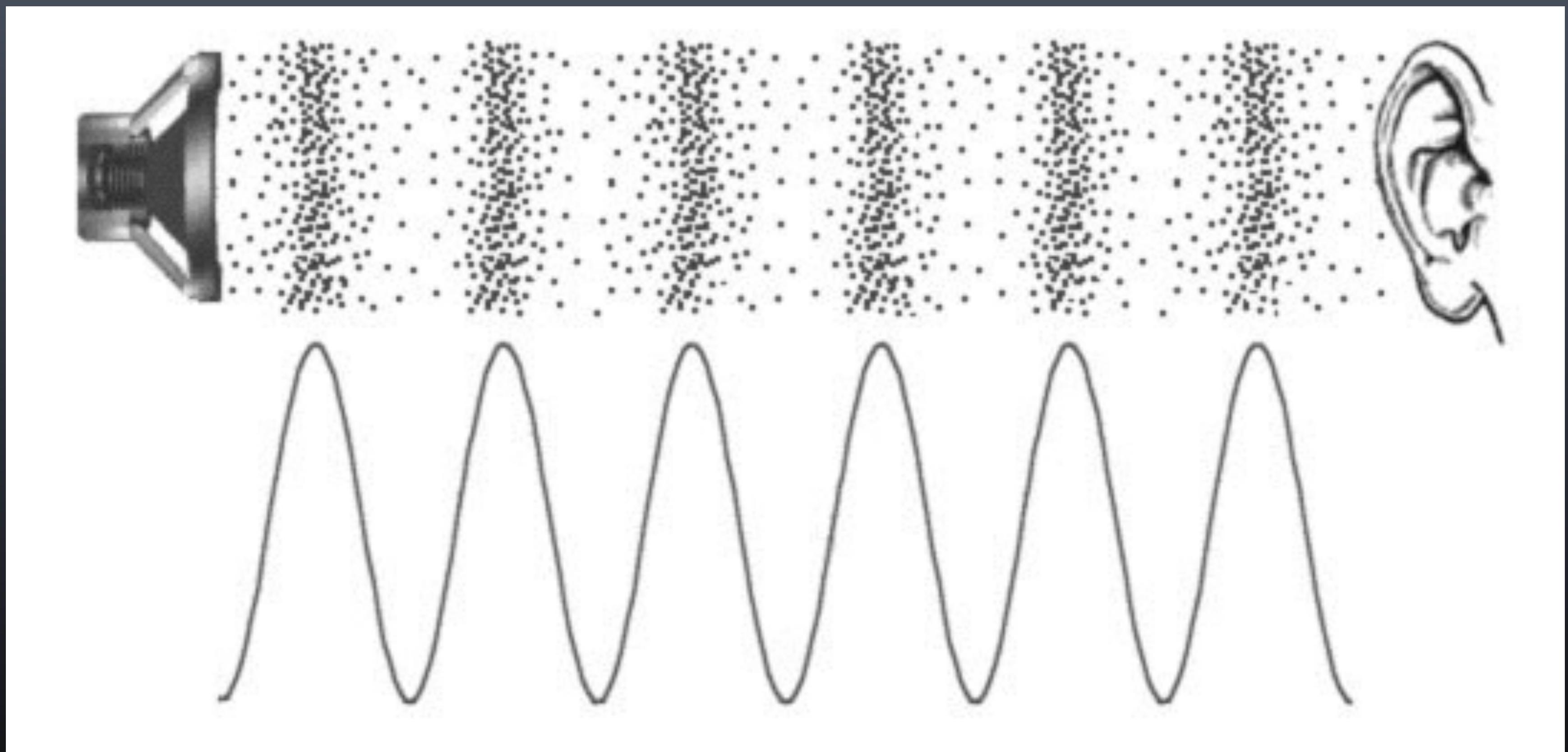
How sound works.

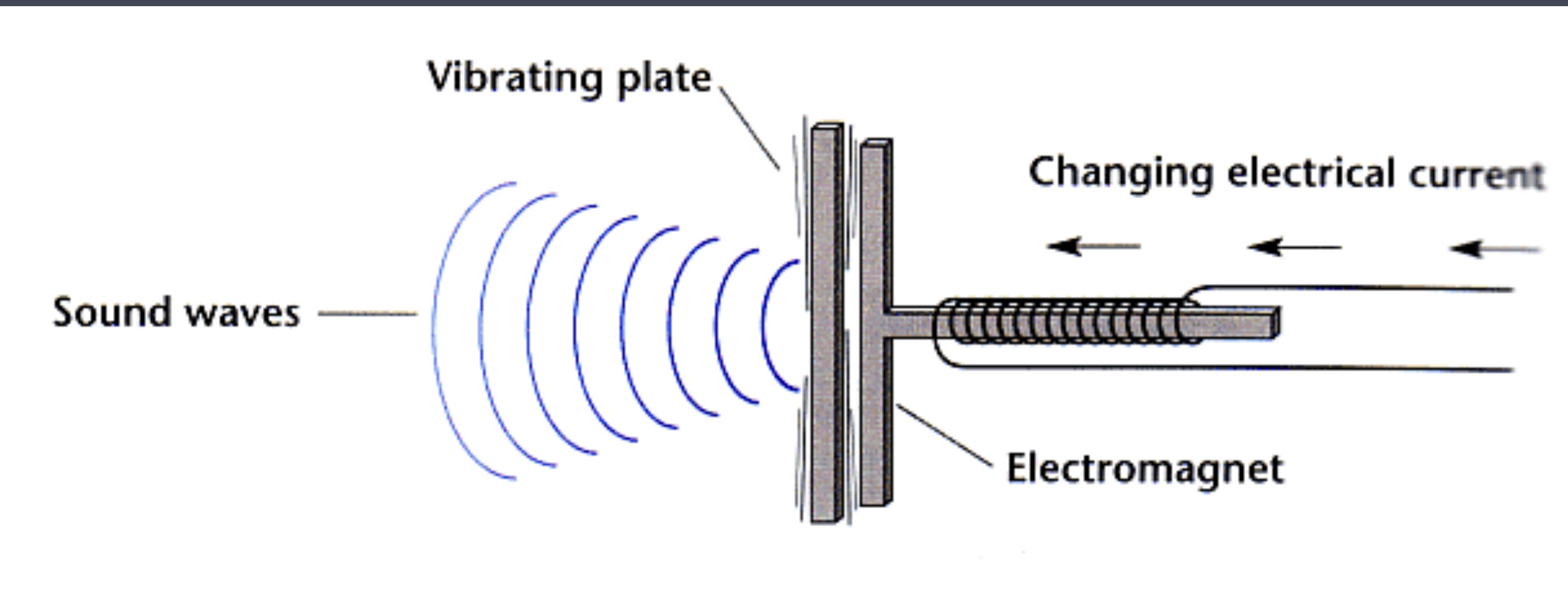




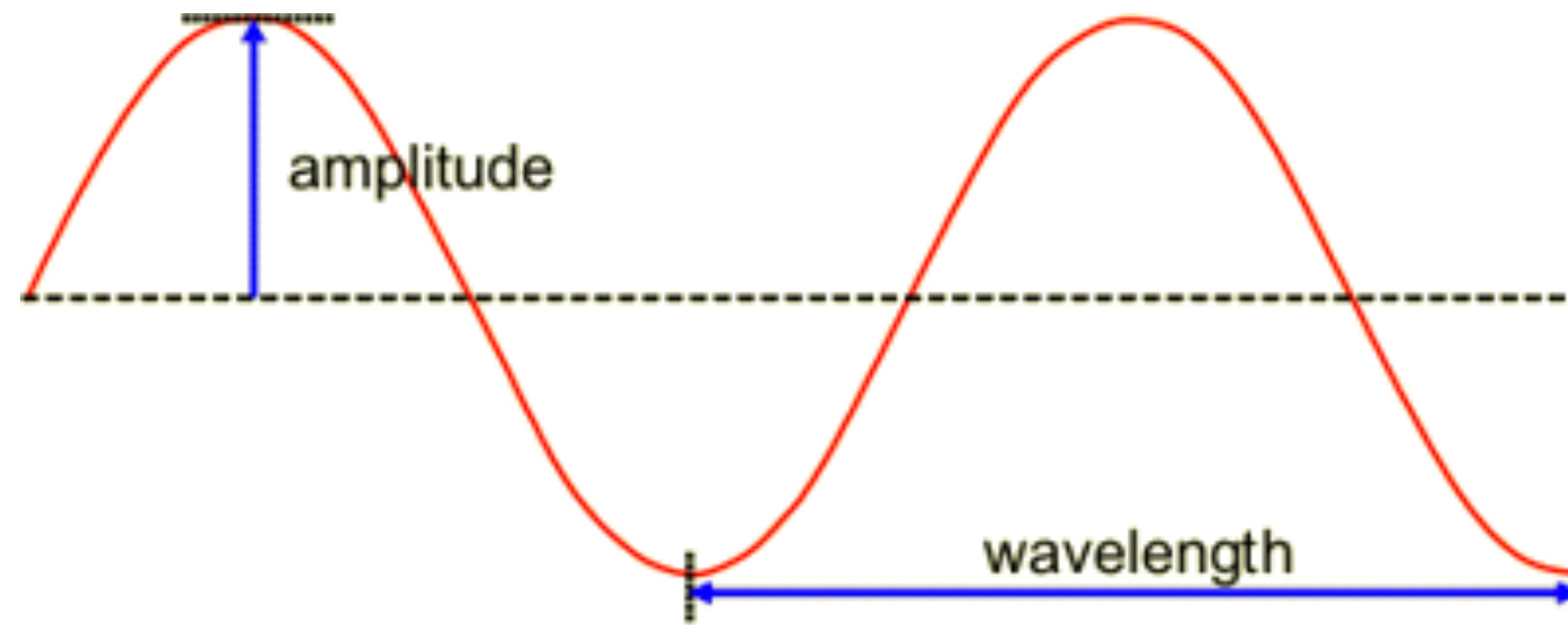
*Figure 1*

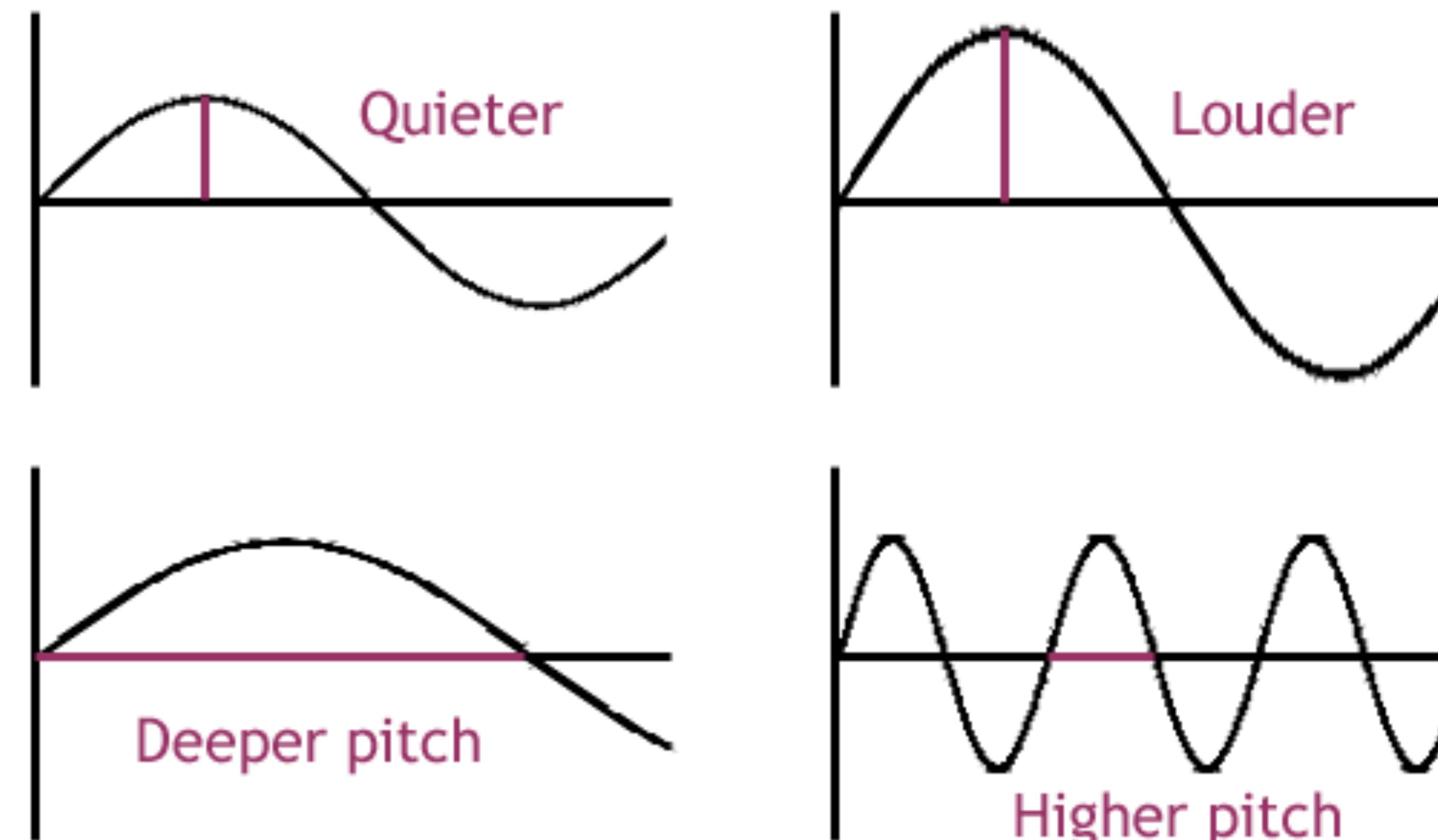






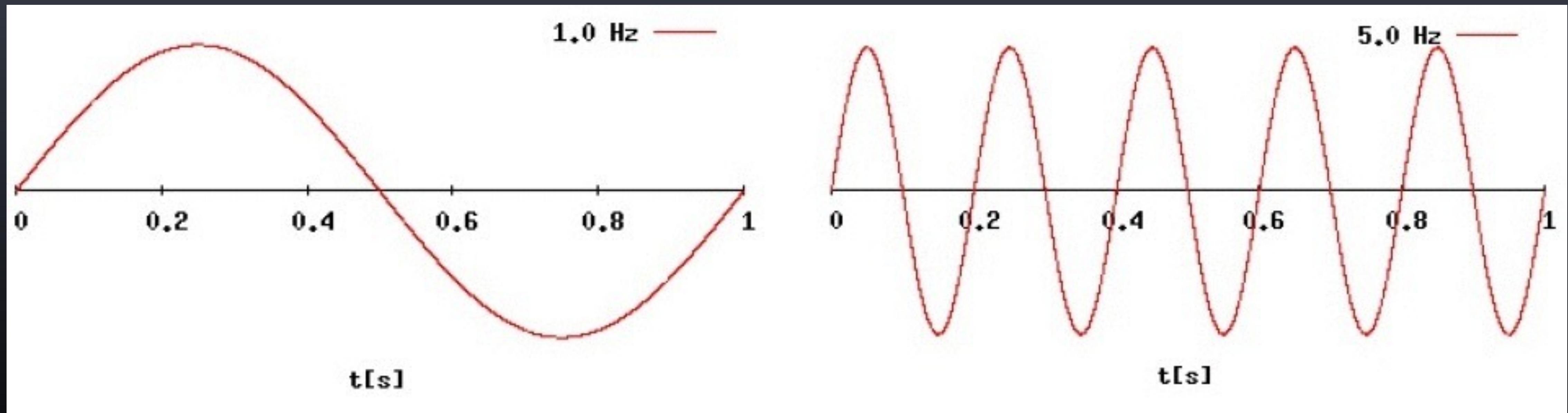
# Sound wave properties.



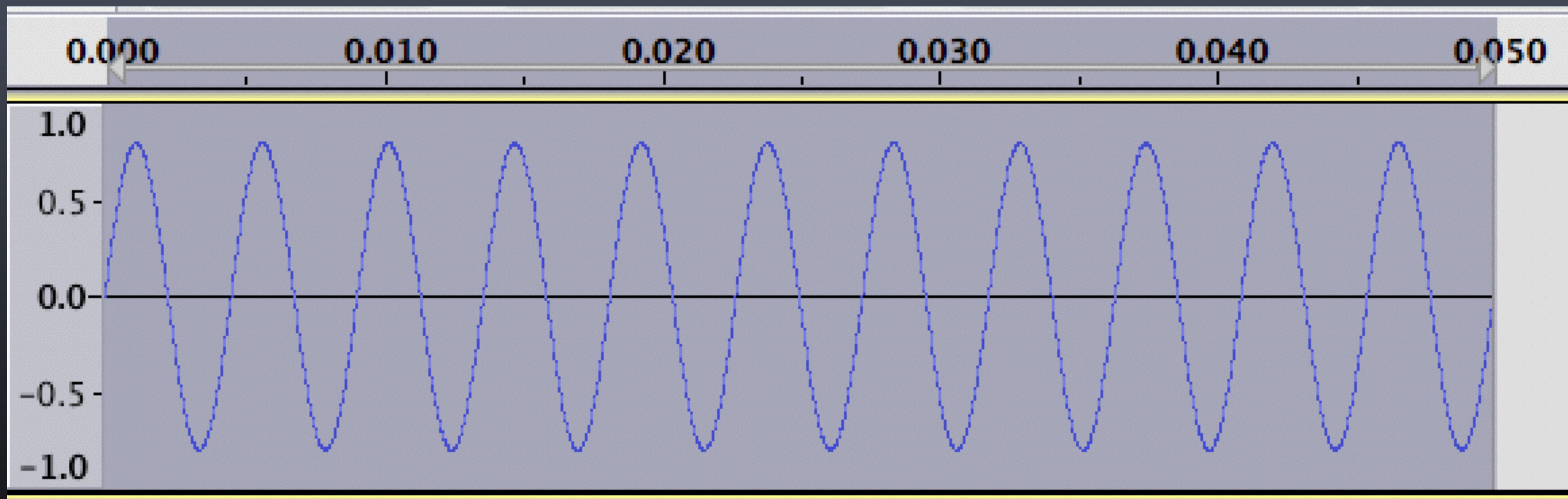


Sound frequency.

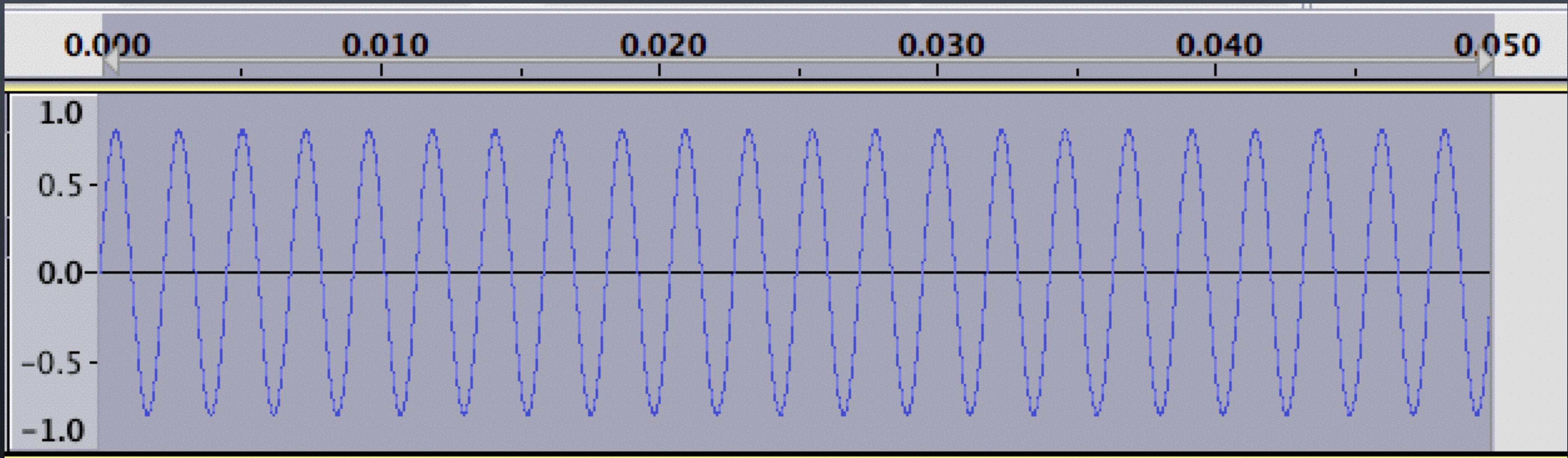
hertz (Hz) = 1 cycle per second.



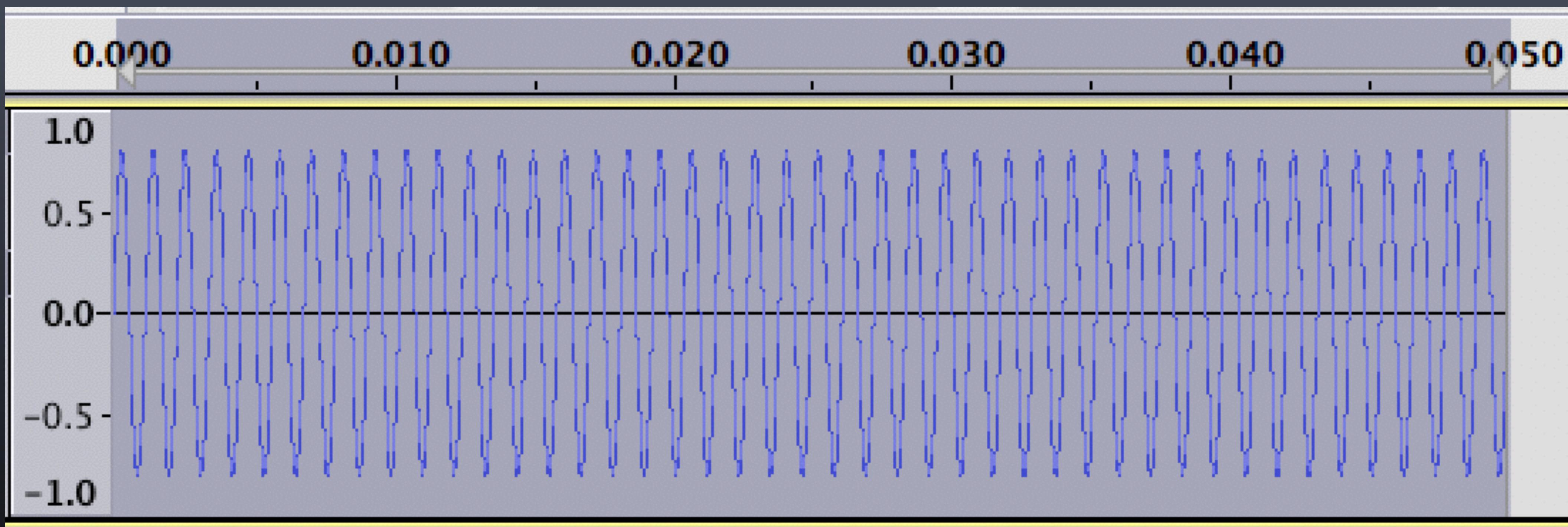
220 Hz



# 440 Hz

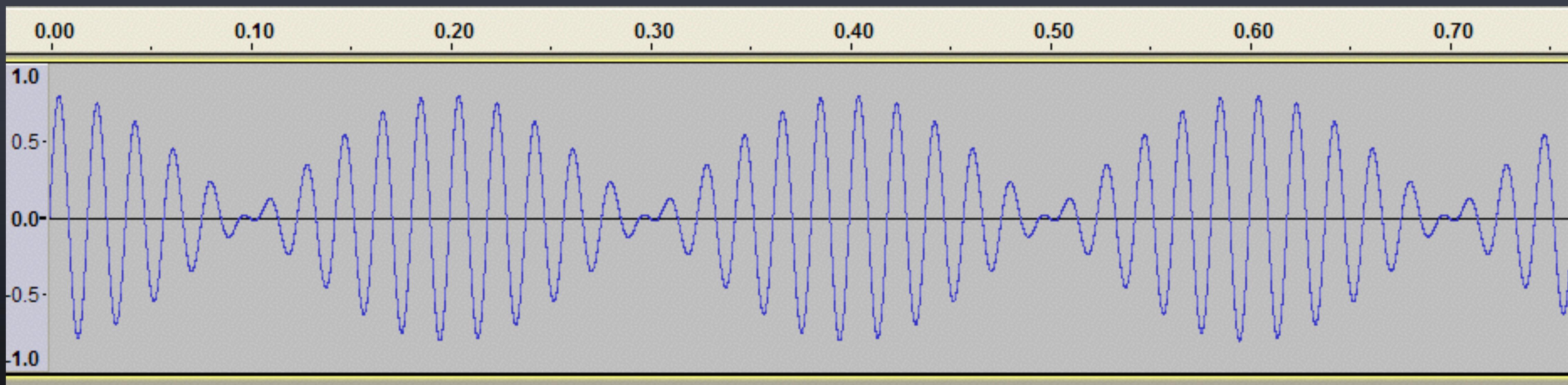


# 880 Hz



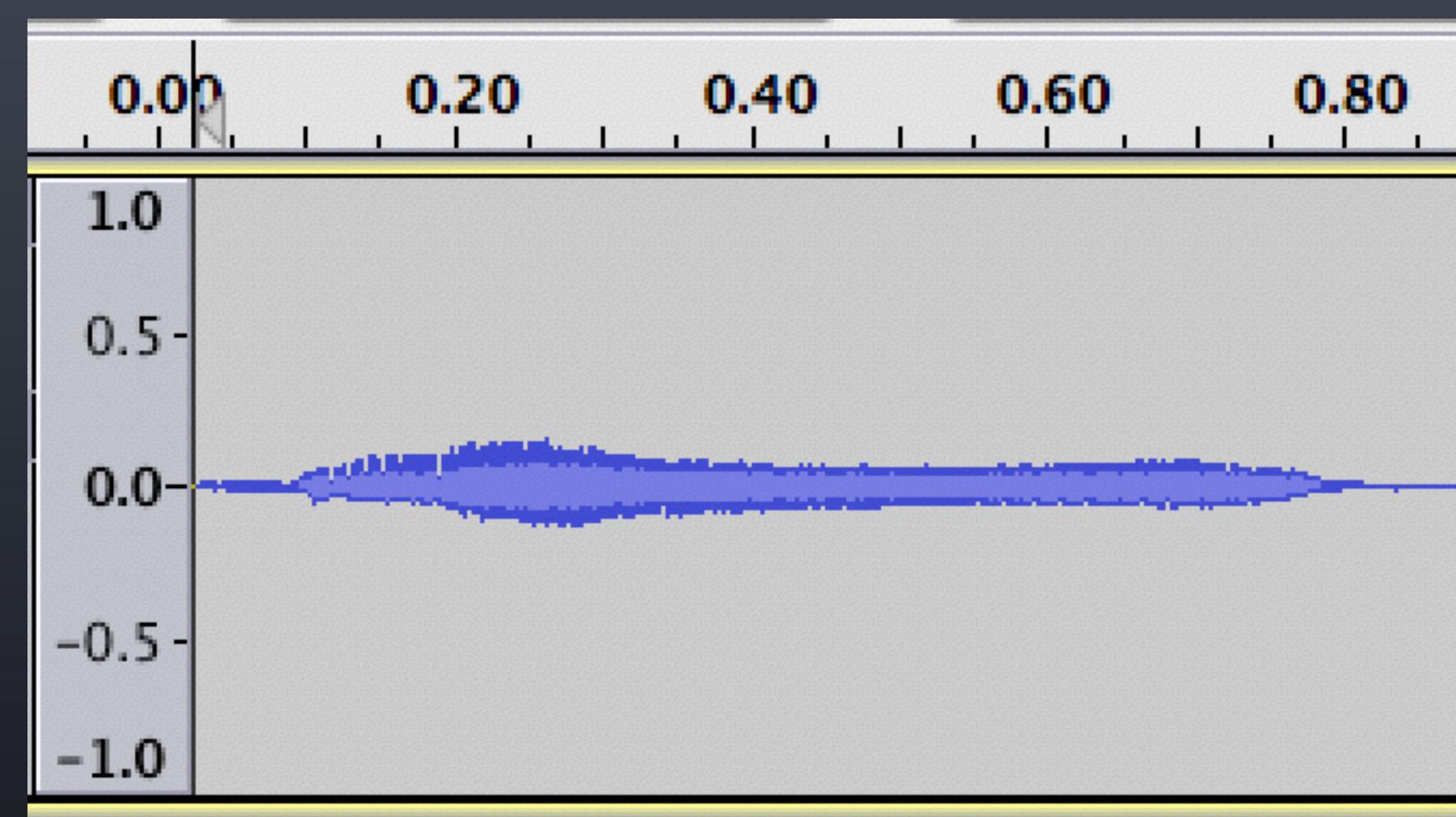
Complex waveforms.

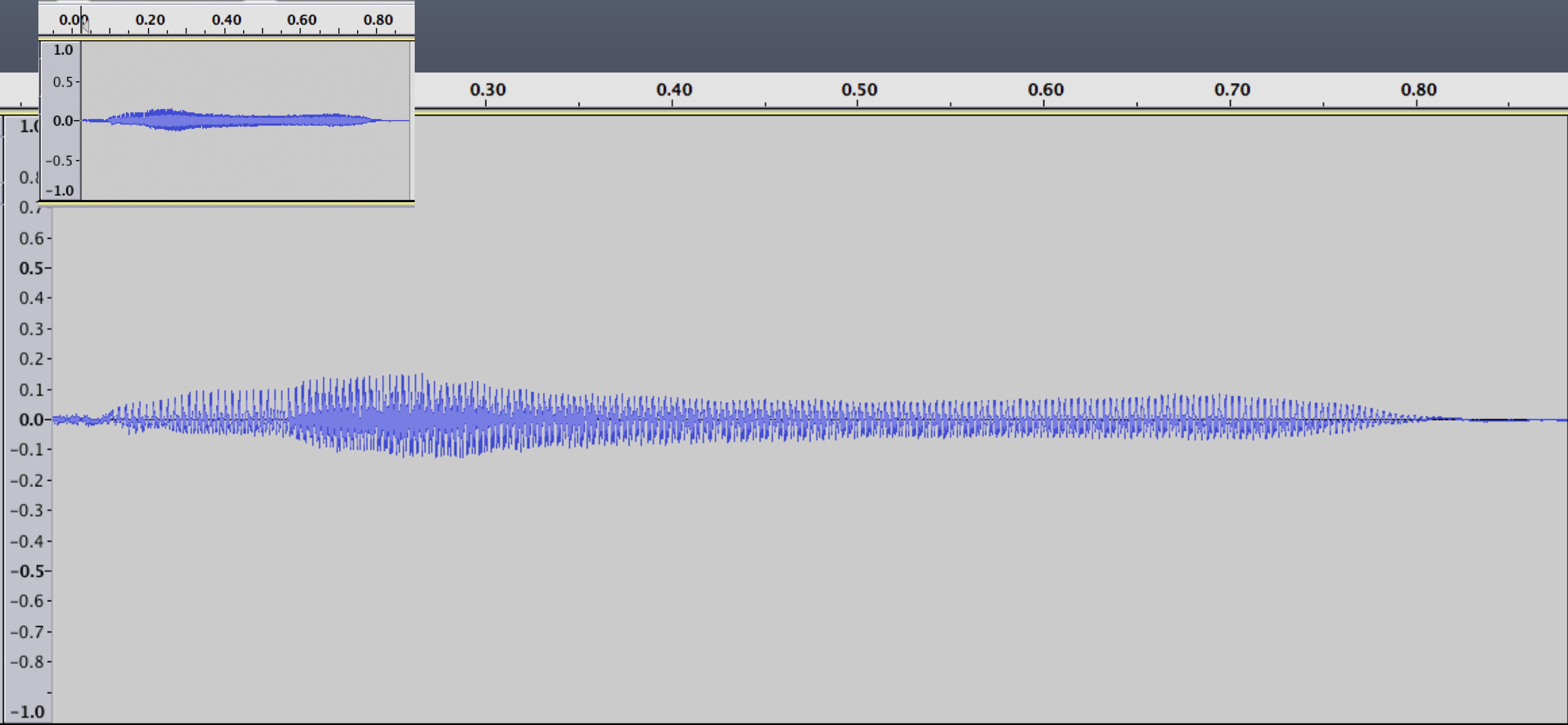
# Modulating frequency.

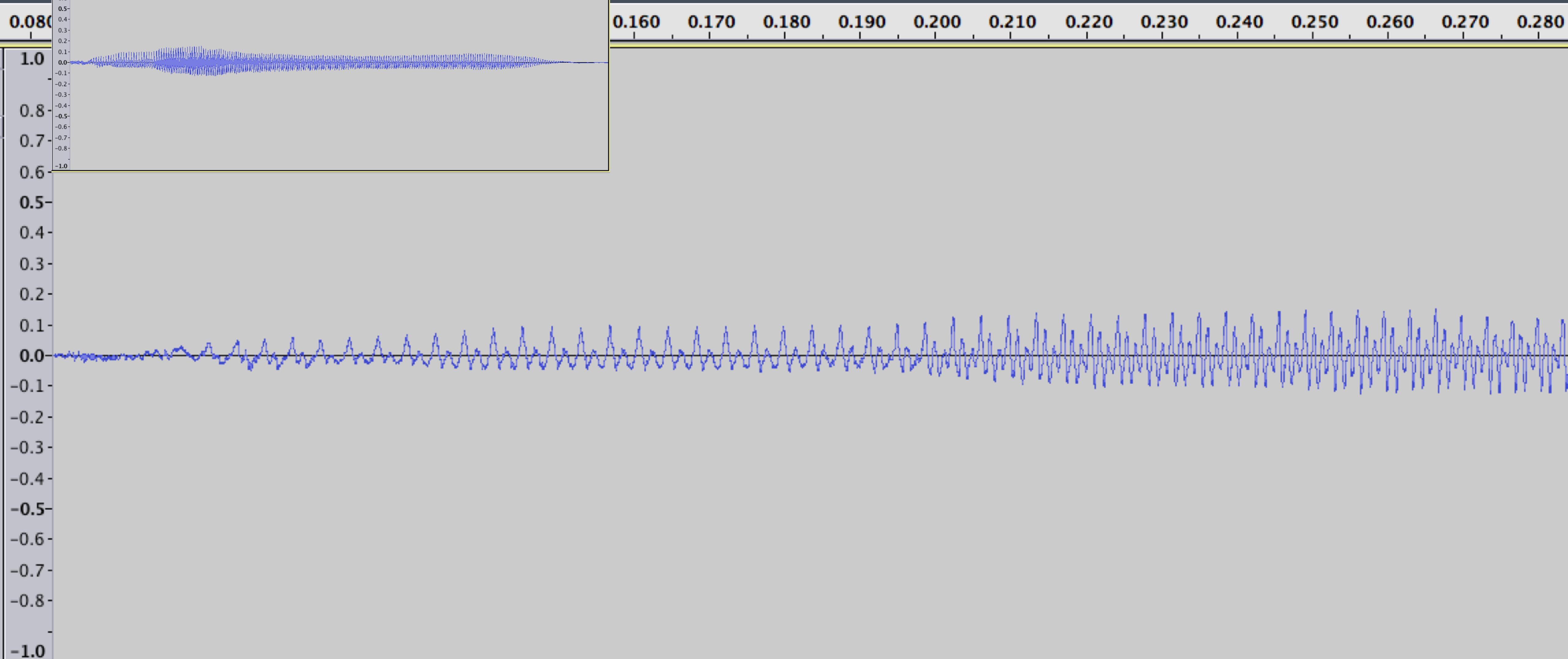
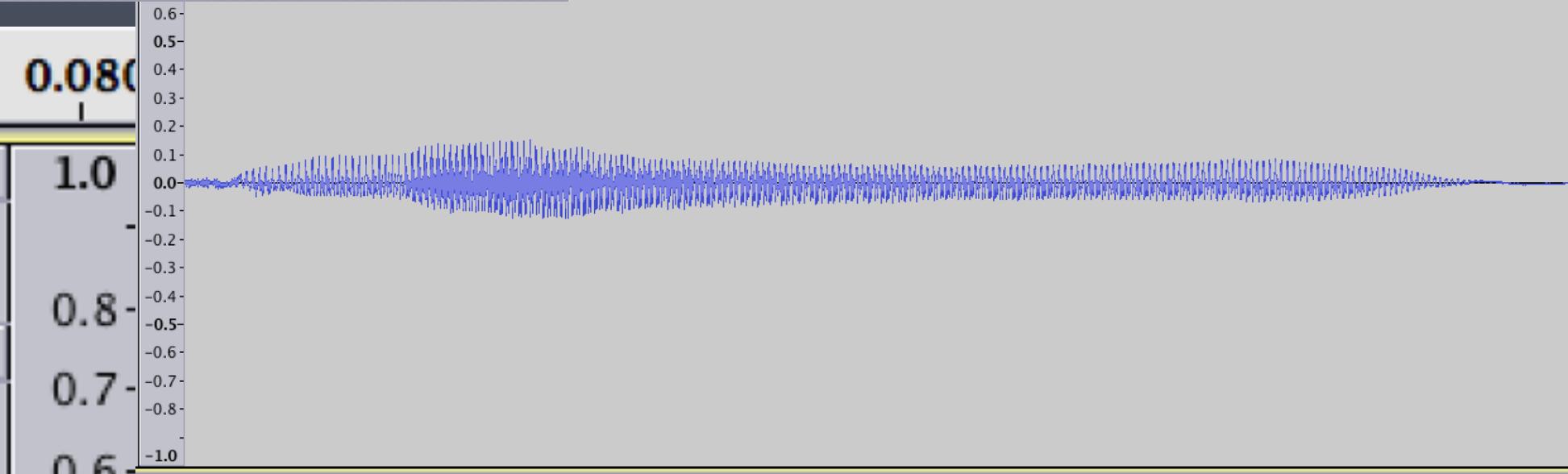
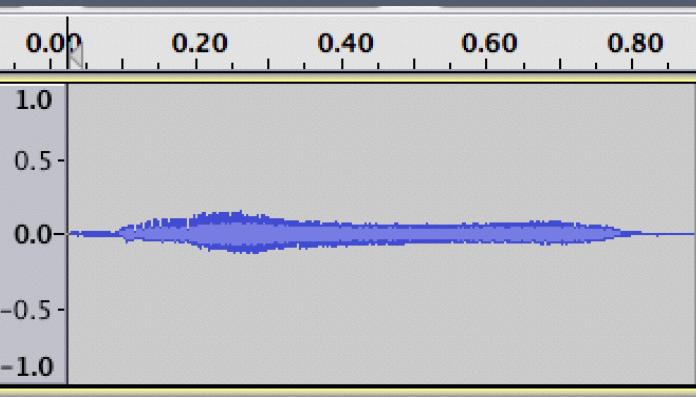


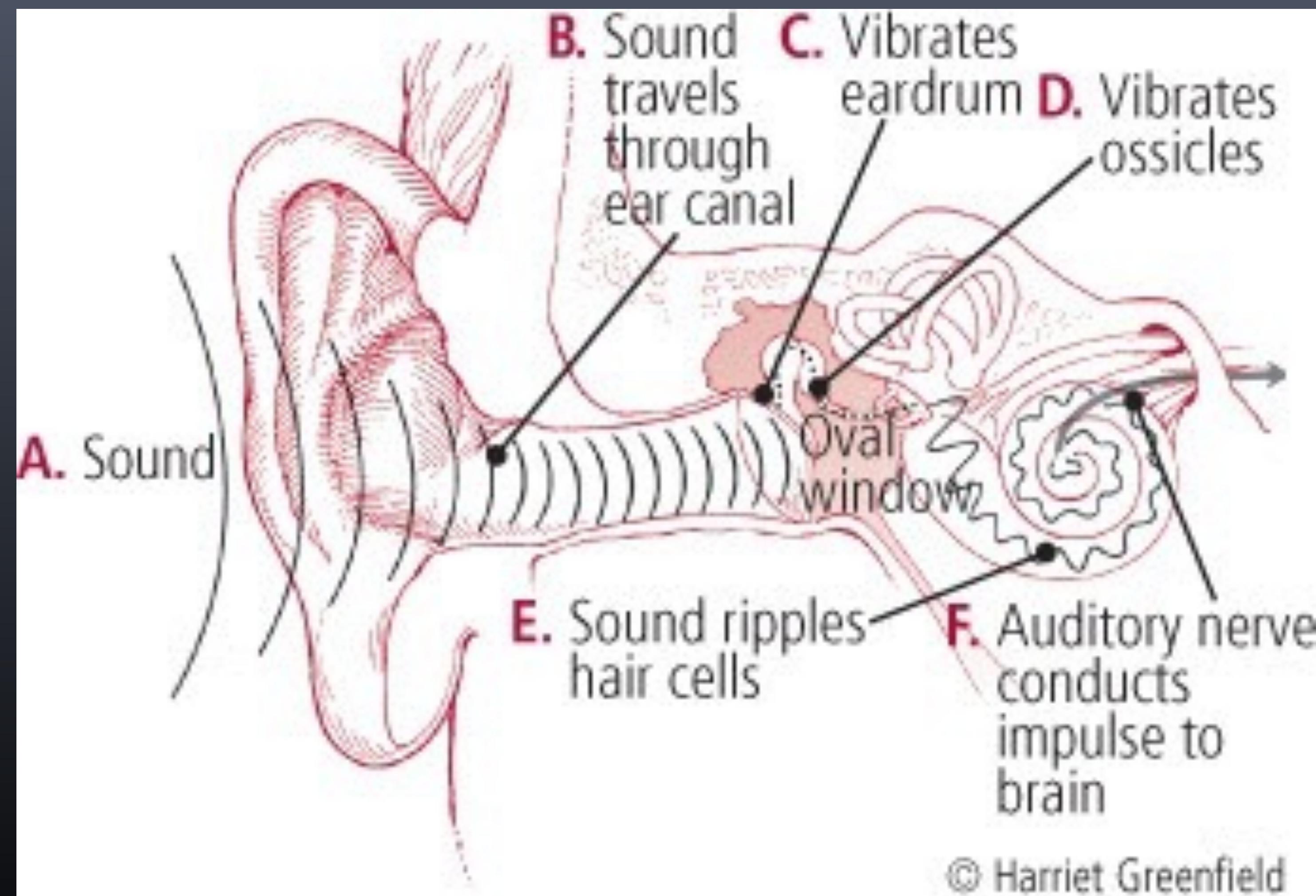


# “Hello”



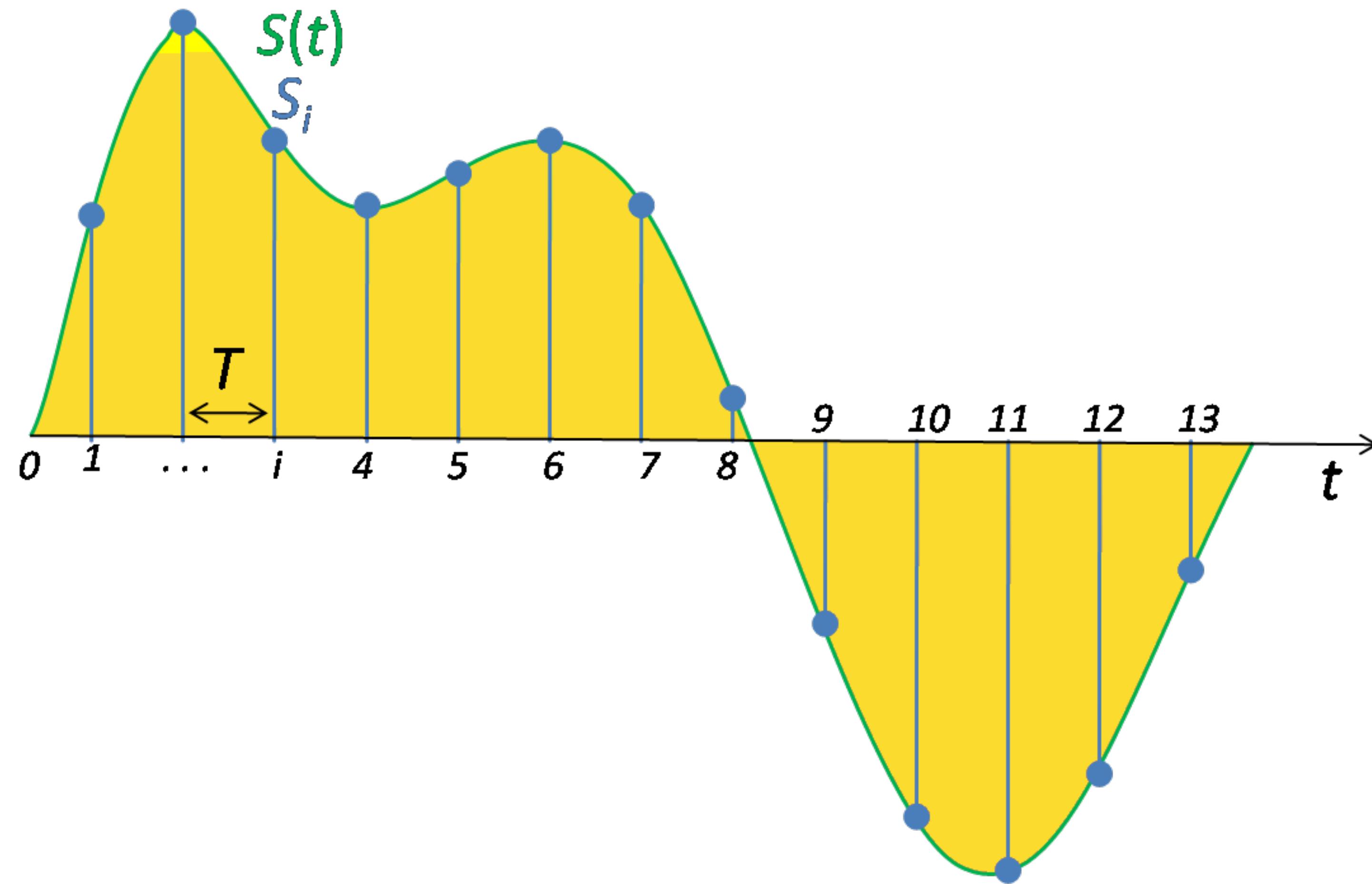






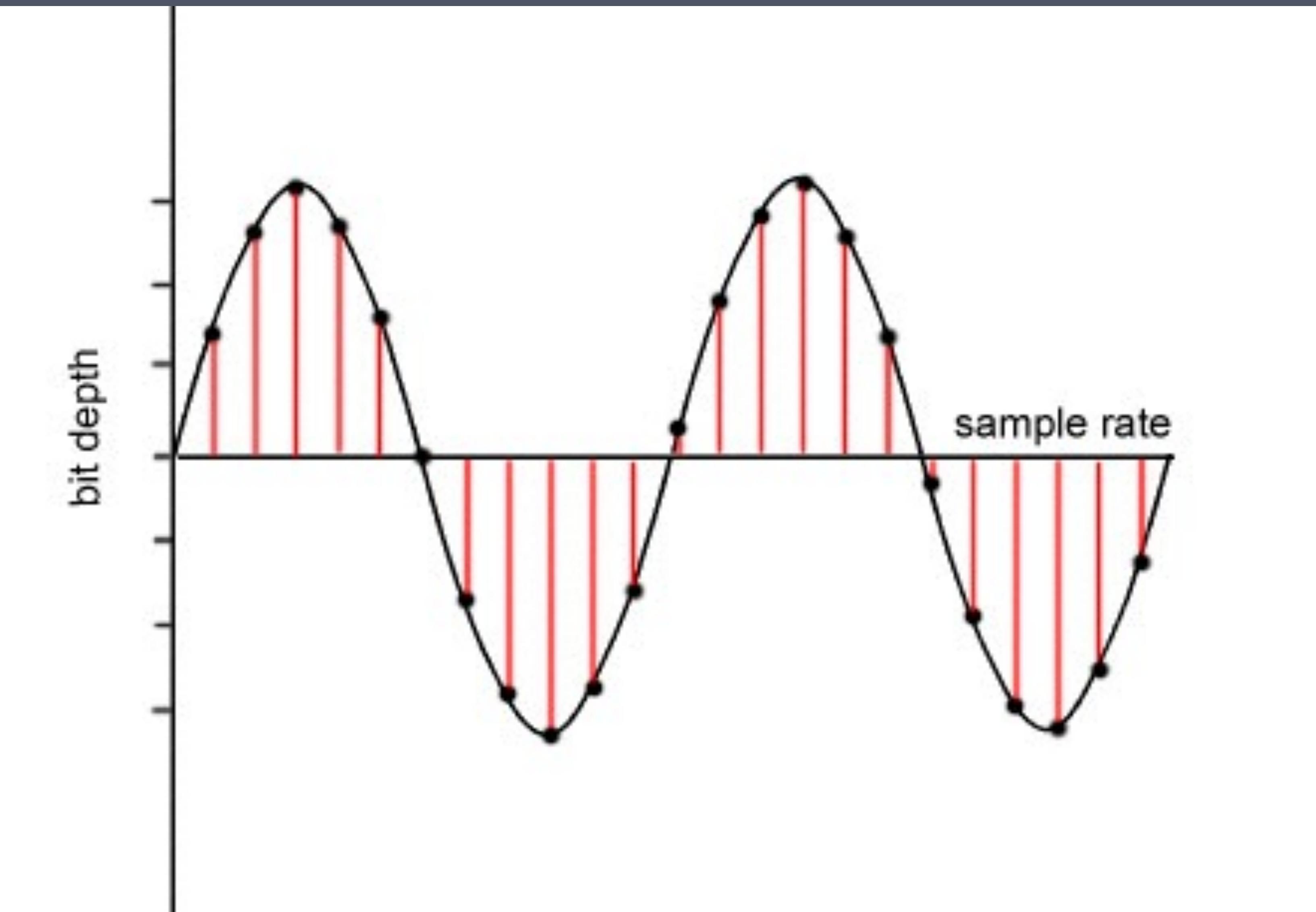
# How digital sound works.

Digital sampling.



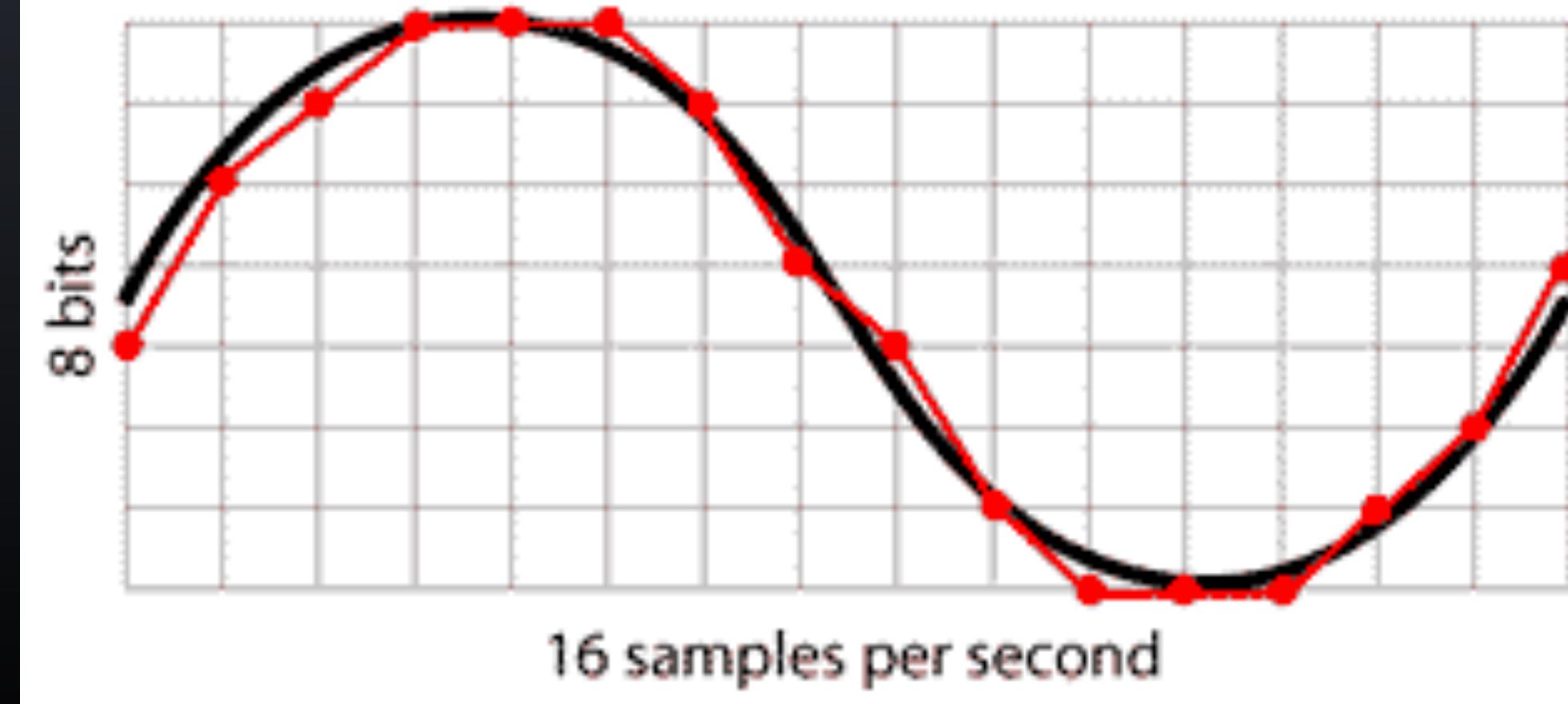
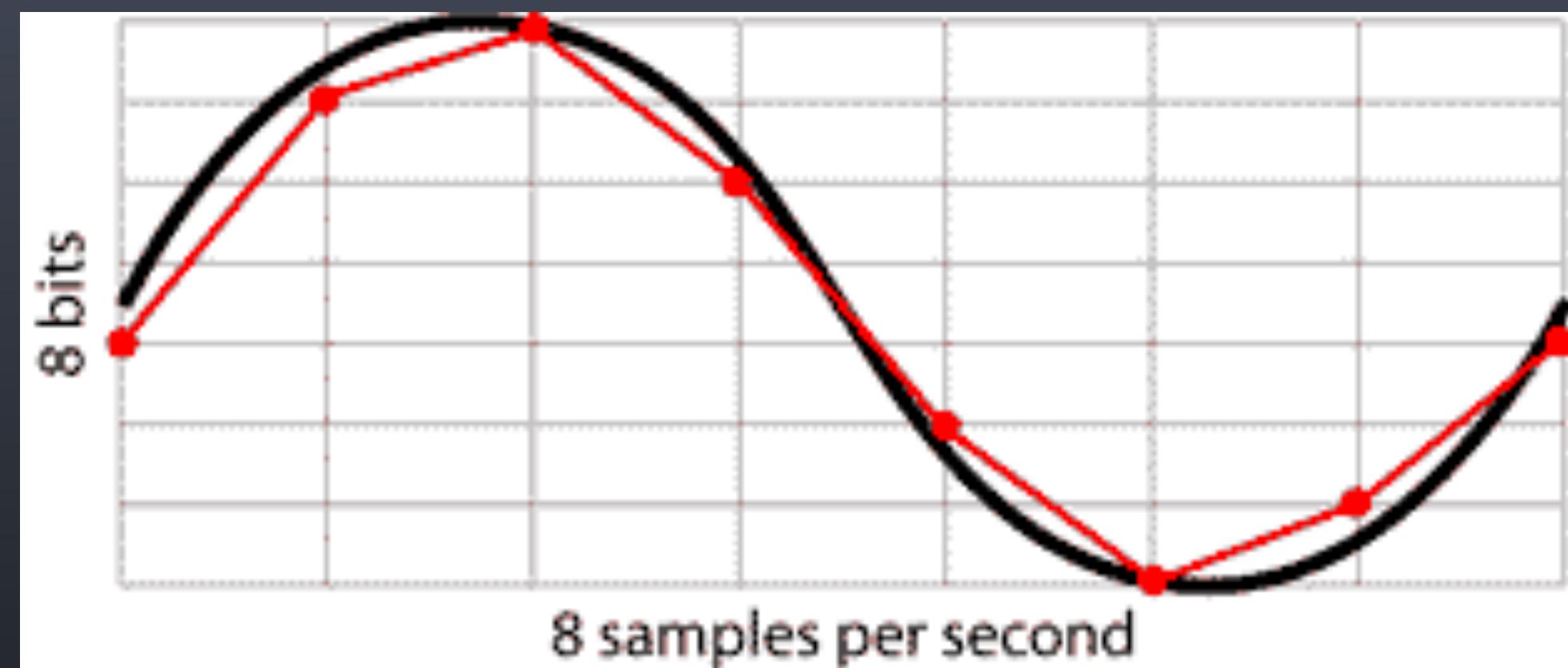
Sampling quality.

Sample rate and bits per sample.



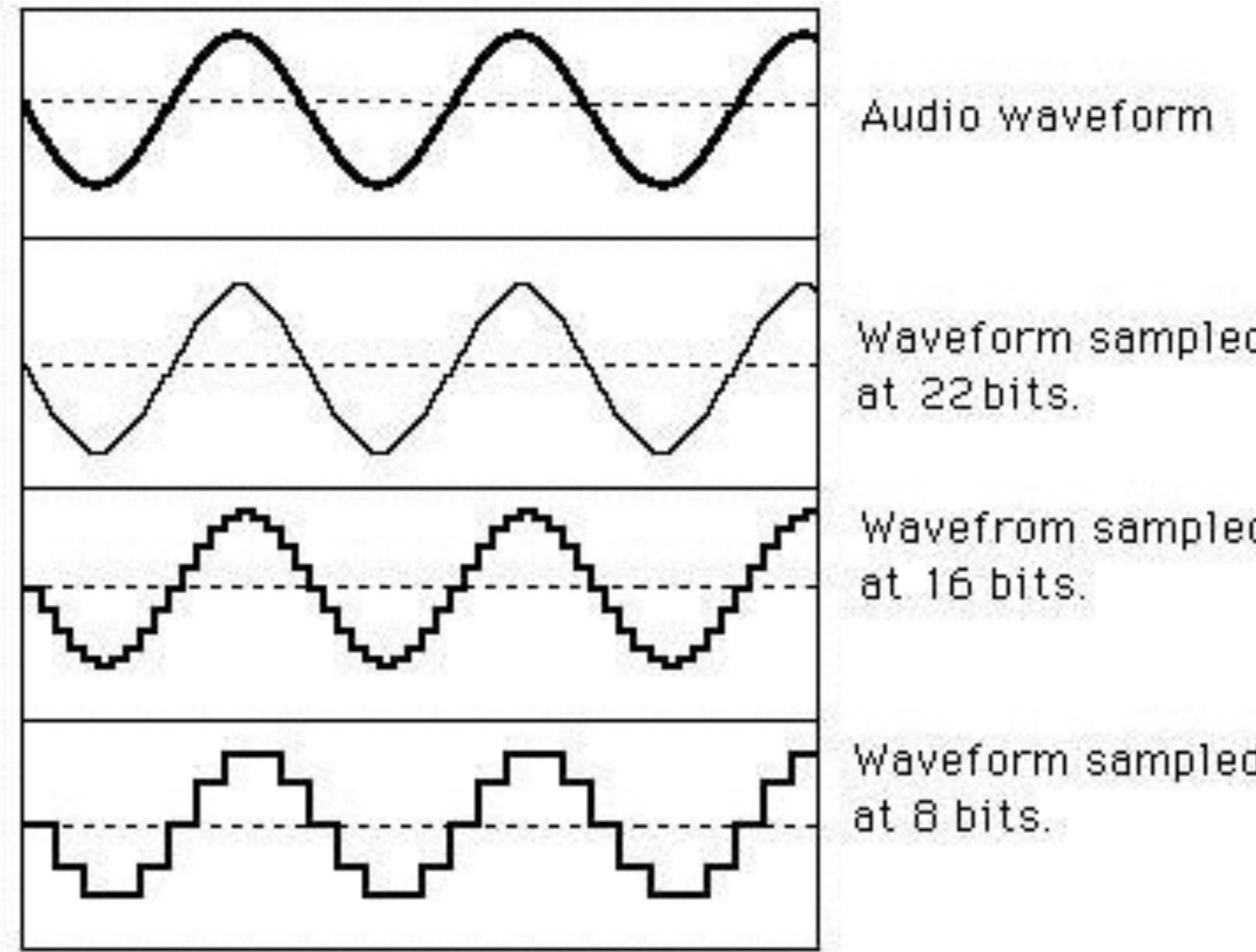
Sample rate.

How many discrete samples each second  
(measured in Hz).



Bits per sample.

## **Sound quality and bits.**



Bitrate.

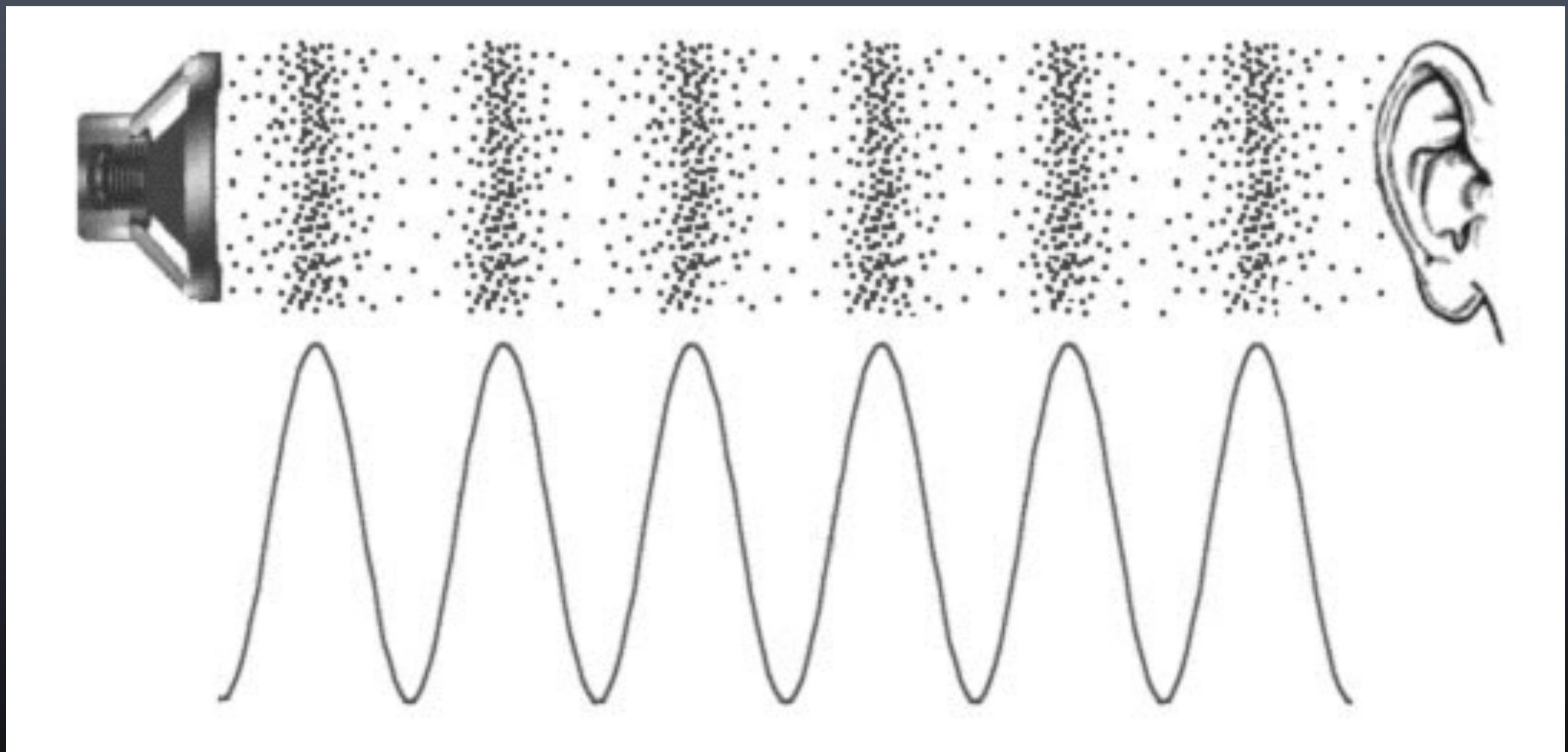
# Bitrate.

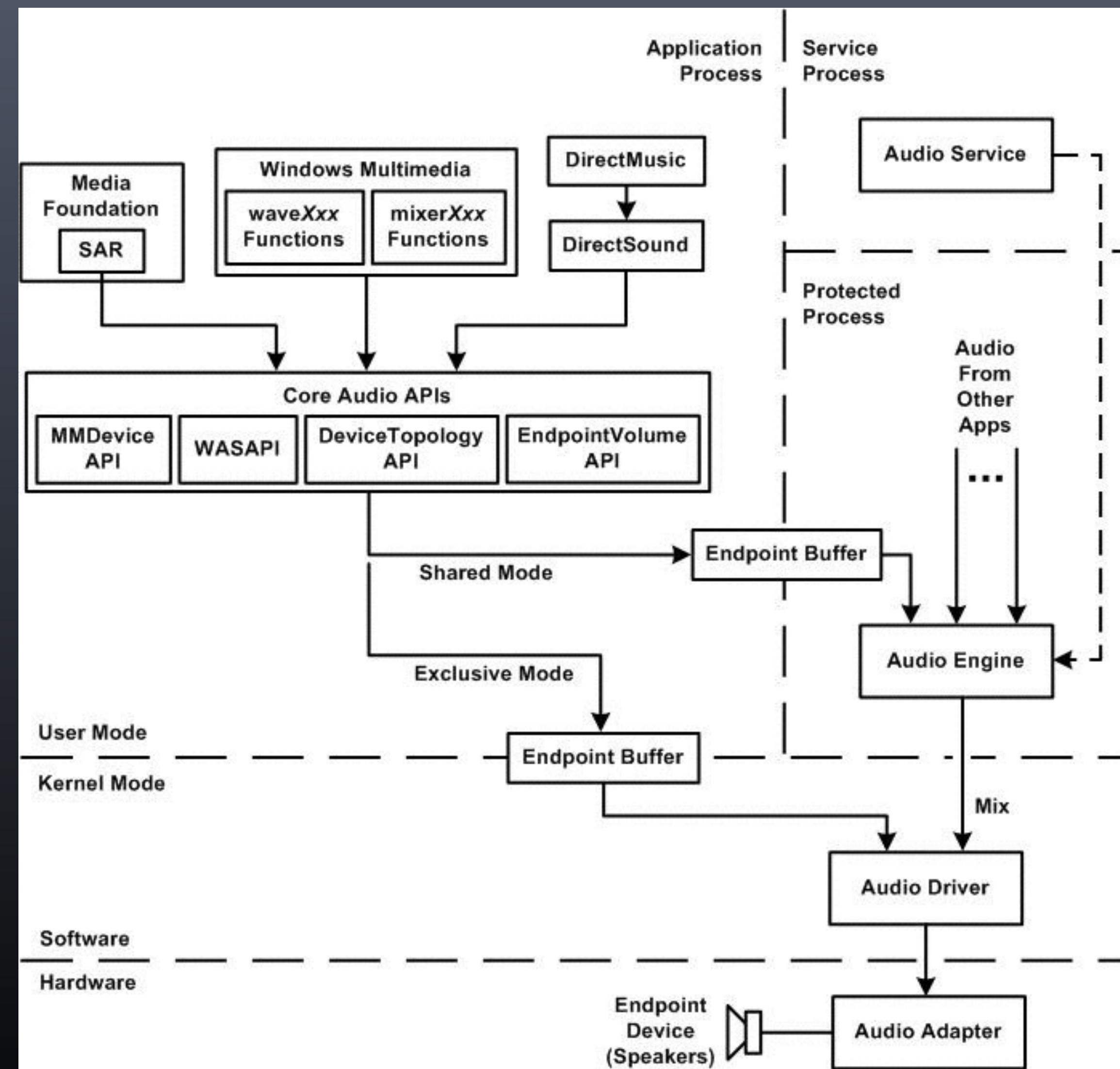
Sampling rate combined with bit depth.

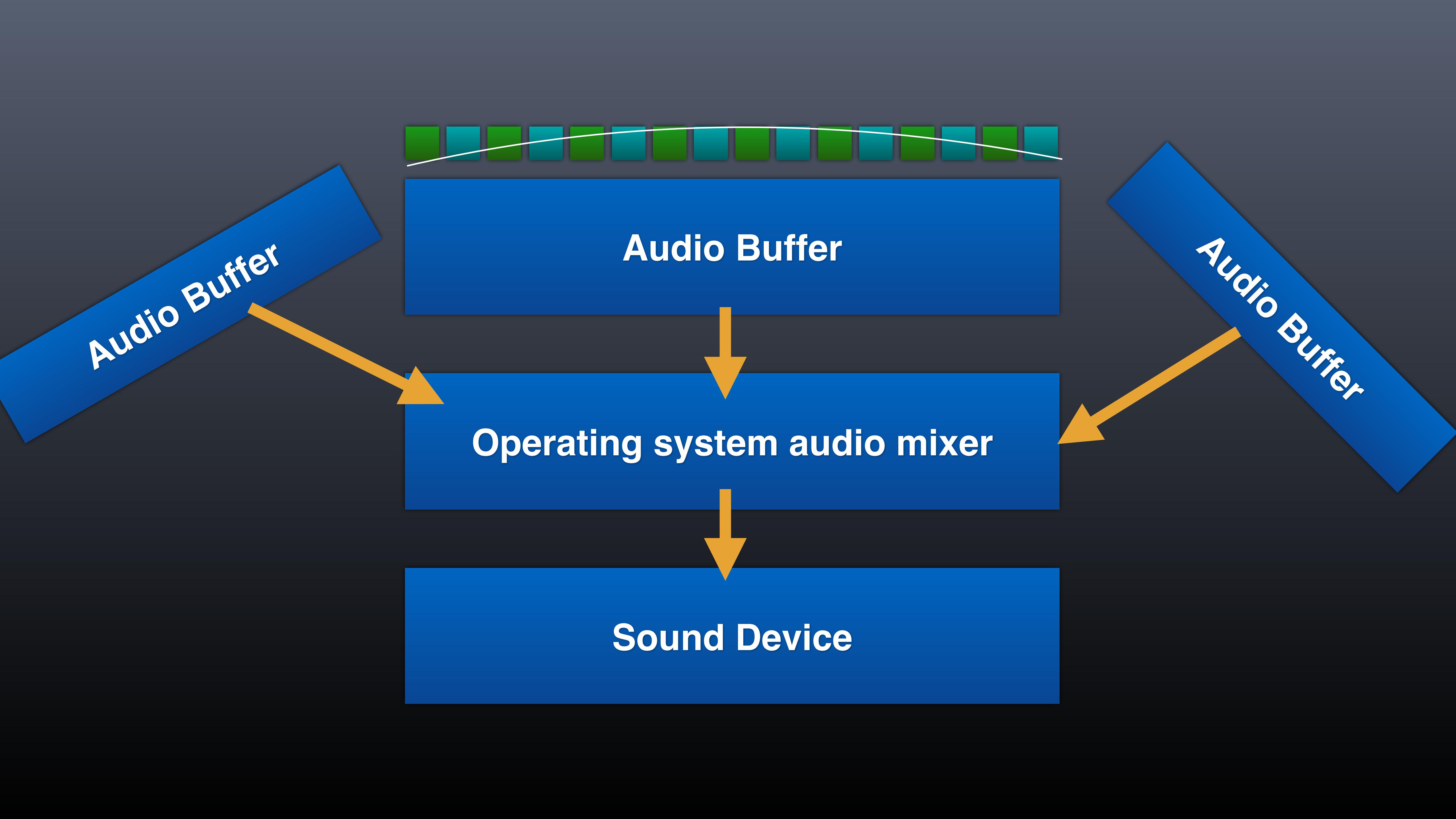
Reflects the total bit count processed per second.

Measured in bits per second or bit/s (or bps)

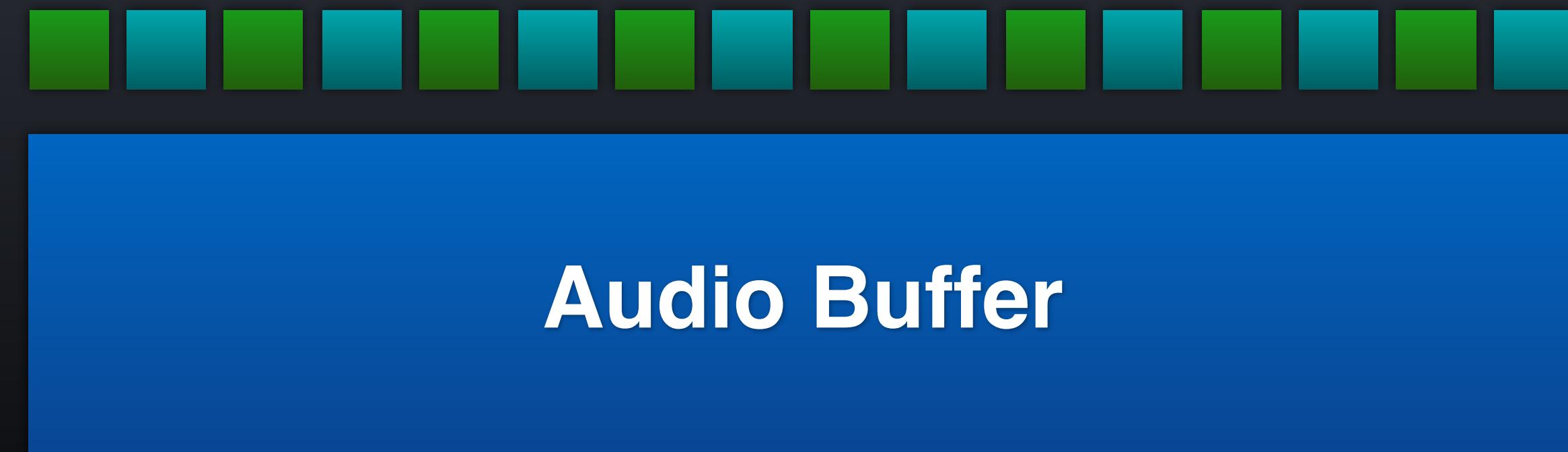
Playing digital audio.







Playing digital sound involves filling an audio buffer with audio data on demand.



# Playing audio with SDL2

SDL\_mixer (the easy way).

# Download SDL\_mixer.

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

## **Development Libraries:**

Windows

[SDL2\\_mixer-devel-2.0.0-VC.zip](#) (Visual C++ 32/64-bit)

[SDL2\\_mixer-devel-2.0.0-mingw.tar.gz](#) (MinGW 32/64-bit)

Mac OS X

[SDL2\\_mixer-2.0.0.dmg](#) (Intel 10.5+)

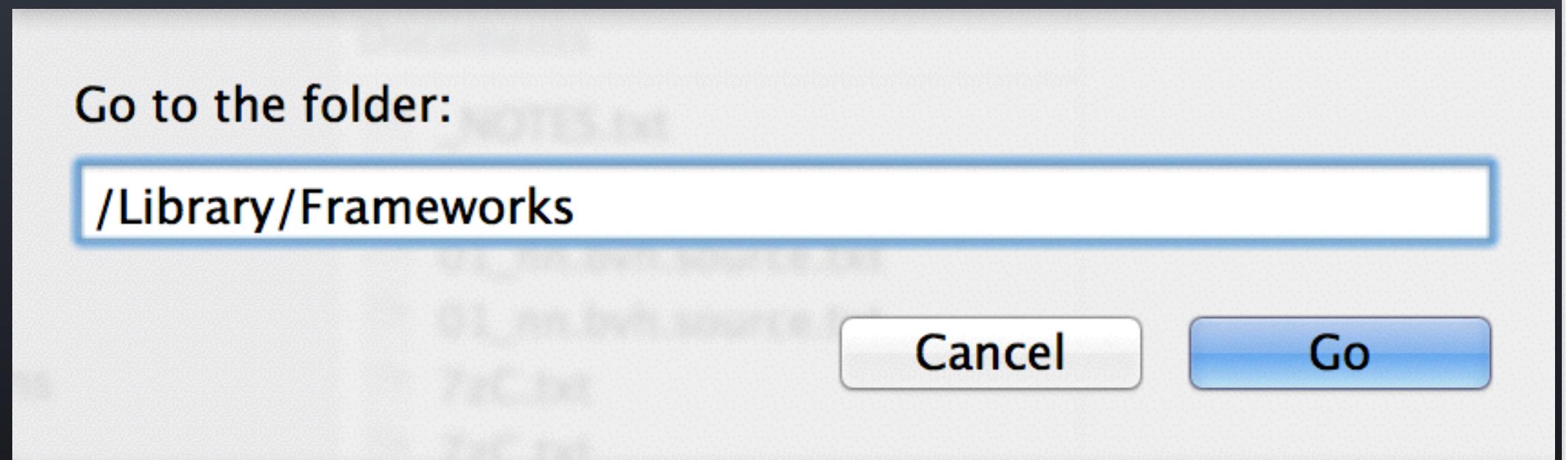
Linux

Please contact your distribution maintainer for updates.

iOS & Android

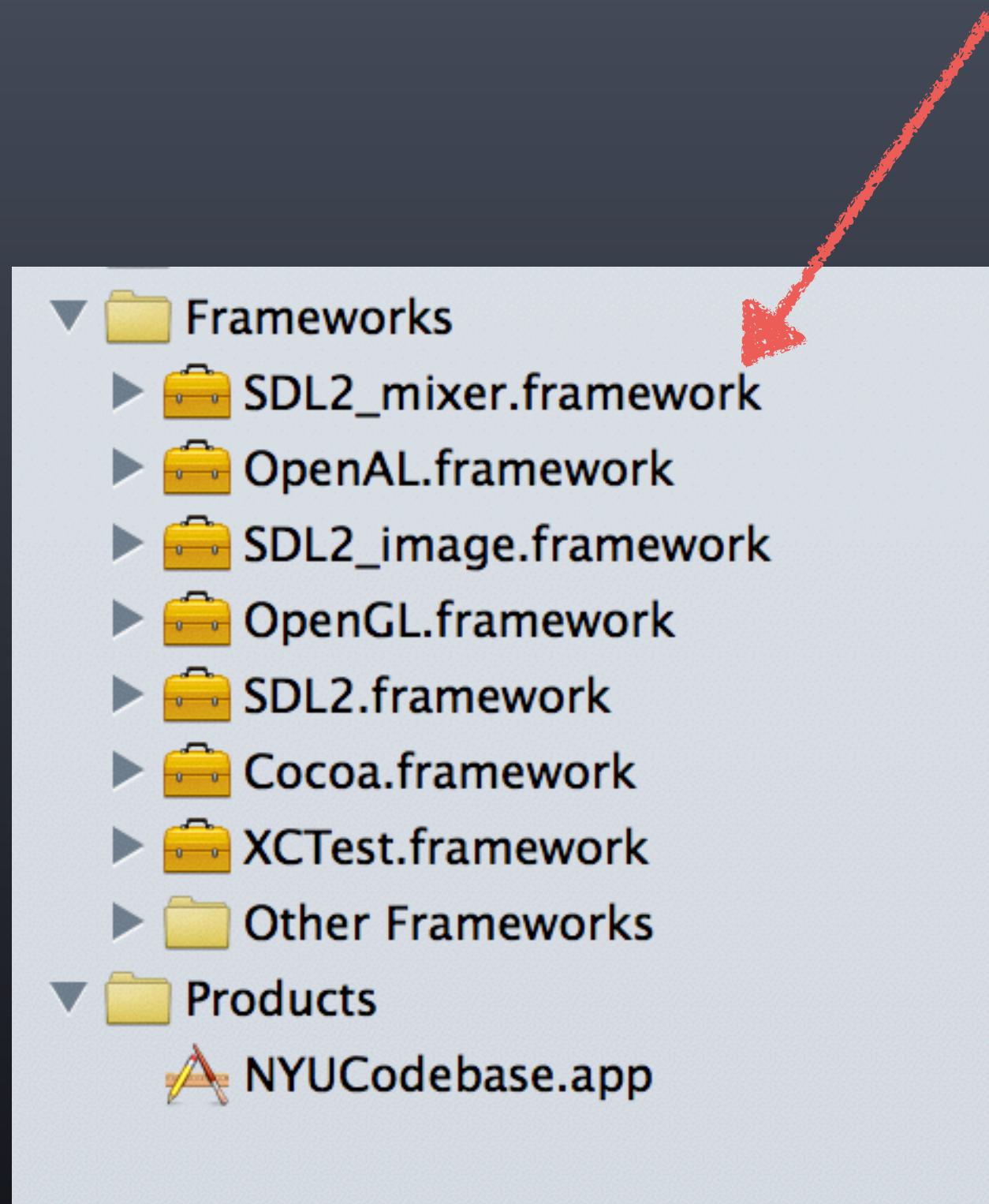
Projects for these platforms are included with the [source](#).

On Mac



Application Support	►	Adobe AIR.framework
Audio	►	AEProfiling.framework
Bundles	►	AERegistration.framework
Caches	►	AudioMixEngine.framework
ColorPickers	►	iTunesLibrary.framework
ColorSync	►	Mono.framework
Components	►	NyxAudioAnalysis.framework
Compositions	►	PluginManager.framework
Contextual Menu Items	►	SDL.framework
CoreMediaIO	►	SDL2_image.framework
Desktop Pictures	►	SDL2_mixer.framework
Developer	►	SDL2.framework
Dictionaries	►	WacomMultiTouch.framework
DirectoryServices	►	
Documentation	►	
DropboxHelperTools	►	
Extensions	►	
Filesystems	►	
Fonts	►	
Frameworks	►	
Google	►	

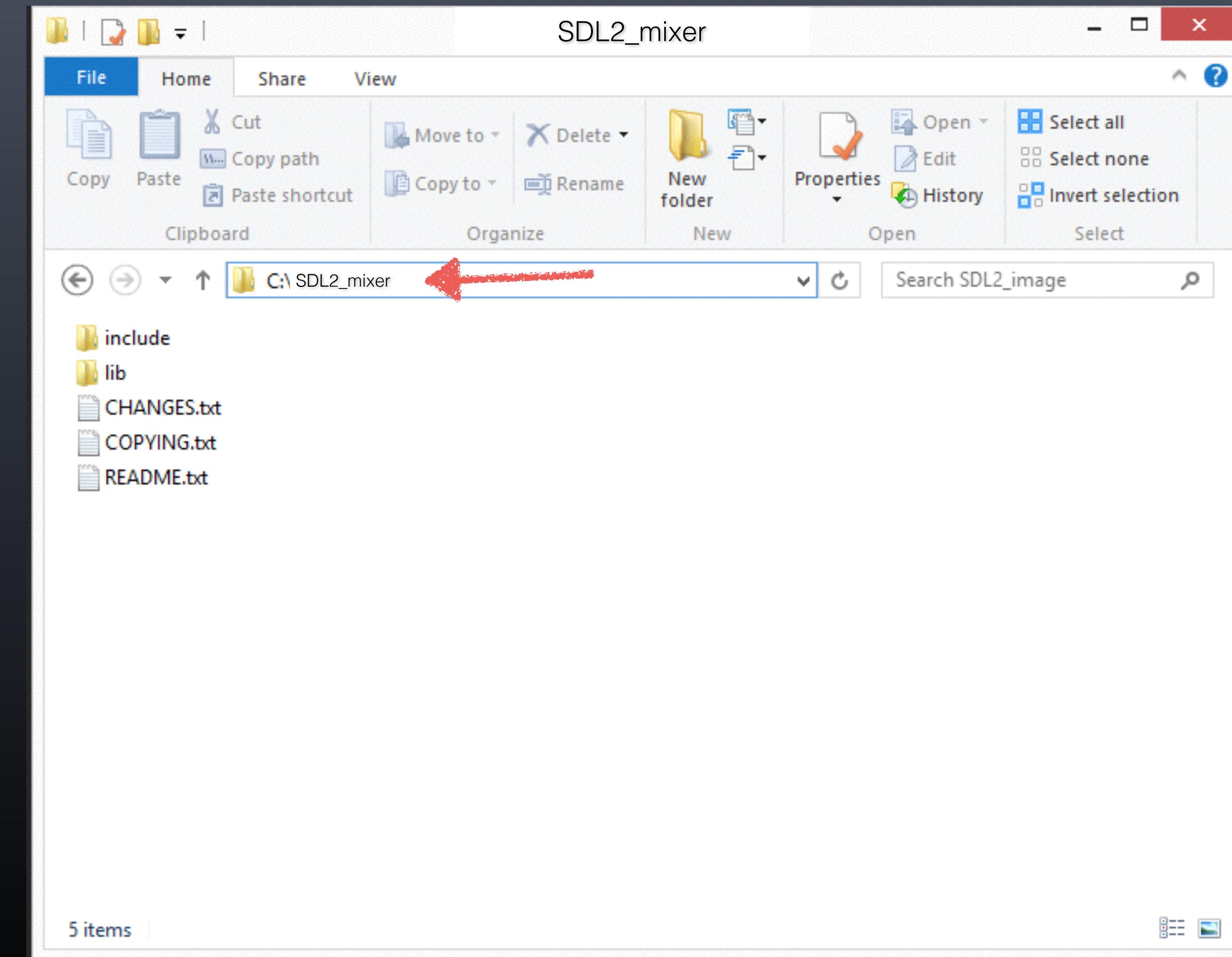
# DRAG FROM FINDER



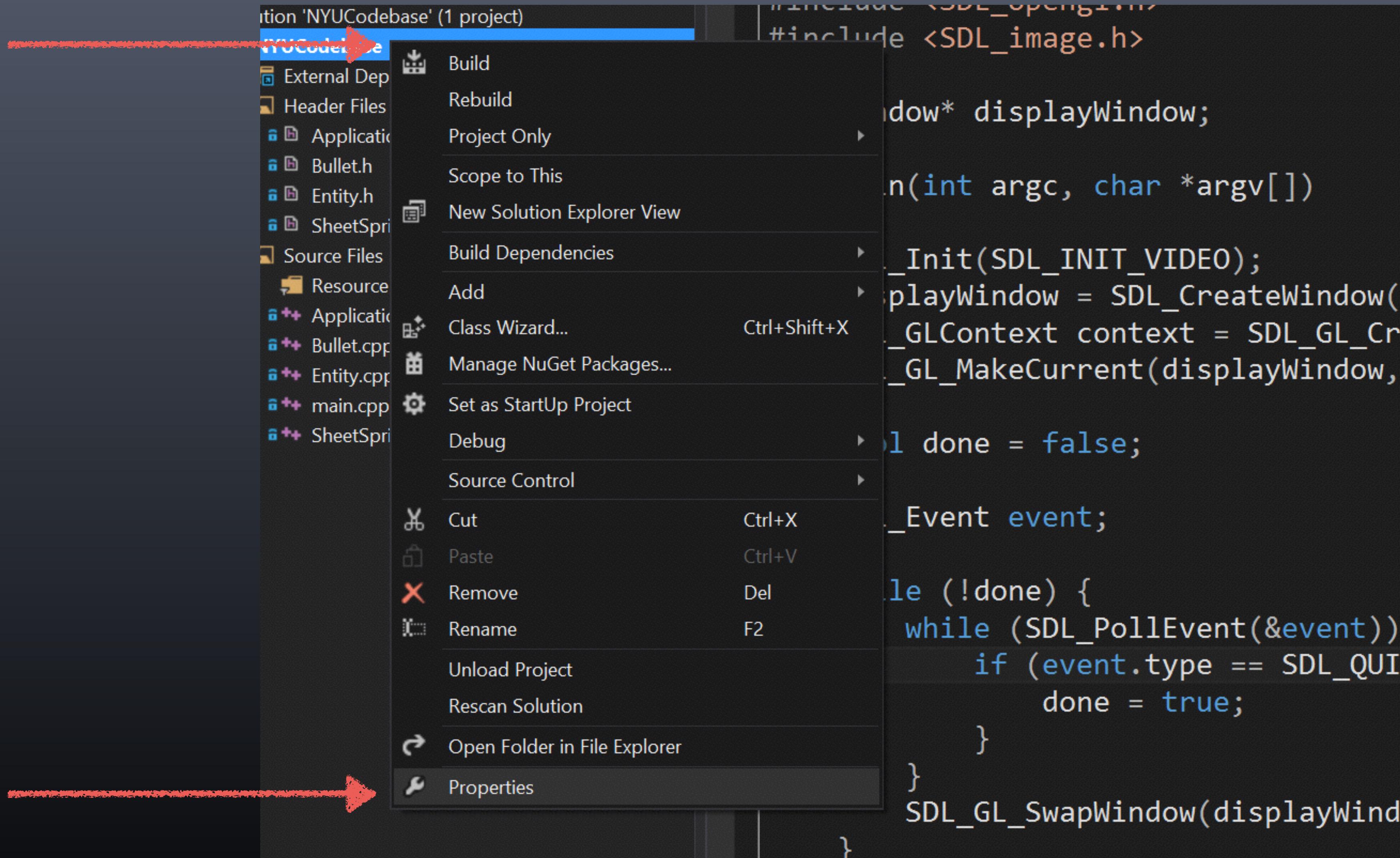
The Xcode Build Settings window for the "NYUCodebase" project shows the "Build Settings" tab selected. In the "Search Paths" section, the "Header Search Paths" table is expanded, showing entries for "Debug" and "Release". A red arrow points to the "Header Search Paths" table in the "Release" section, specifically highlighting the entry for "SDL2\_mixer.framework/Versions/A/Headers".

Setting	Value	Type
Always Search User Paths	No	non-recursive
<b>Framework Search Paths</b>	/Library/Frameworks	non-recursive
<b>Header Search Paths</b>	/Applications/Xcode.app/Contents/Developer/T... /Applications/Xcode.app/Contents/Developer/T... /Applications/Xcode.app/Contents/Developer/T...	non-recursive
<b>Sub-Directories to Exclude in Recur...</b>	\$(inherited)	non-recursive
<b>Sub-Directories to Include in Recur...</b>	/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefau...	non-recursive
<b>User Header Search Paths</b>	/Library/Frameworks/SDL2_image.framework/Versions/A/Headers /Library/Frameworks/SDL2.framework/Versions/A/Headers /Library/Frameworks/SDL2_mixer.framework/Versions/A/Headers	non-recursive
<b>Versioning</b>	Setting	non-recursive
<b>Apple LLVM 5.1 – Language</b>	Setting	non-recursive
C Language Dialect	Yes	non-recursive
Increase Sharing of Precompiled Headers	No	non-recursive
<b>Precompile Prefix Header</b>	Yes	non-recursive
<b>Prefix Header</b>	Yes	non-recursive
Use Standard System Header Directory Searching	Yes	non-recursive

# On Windows

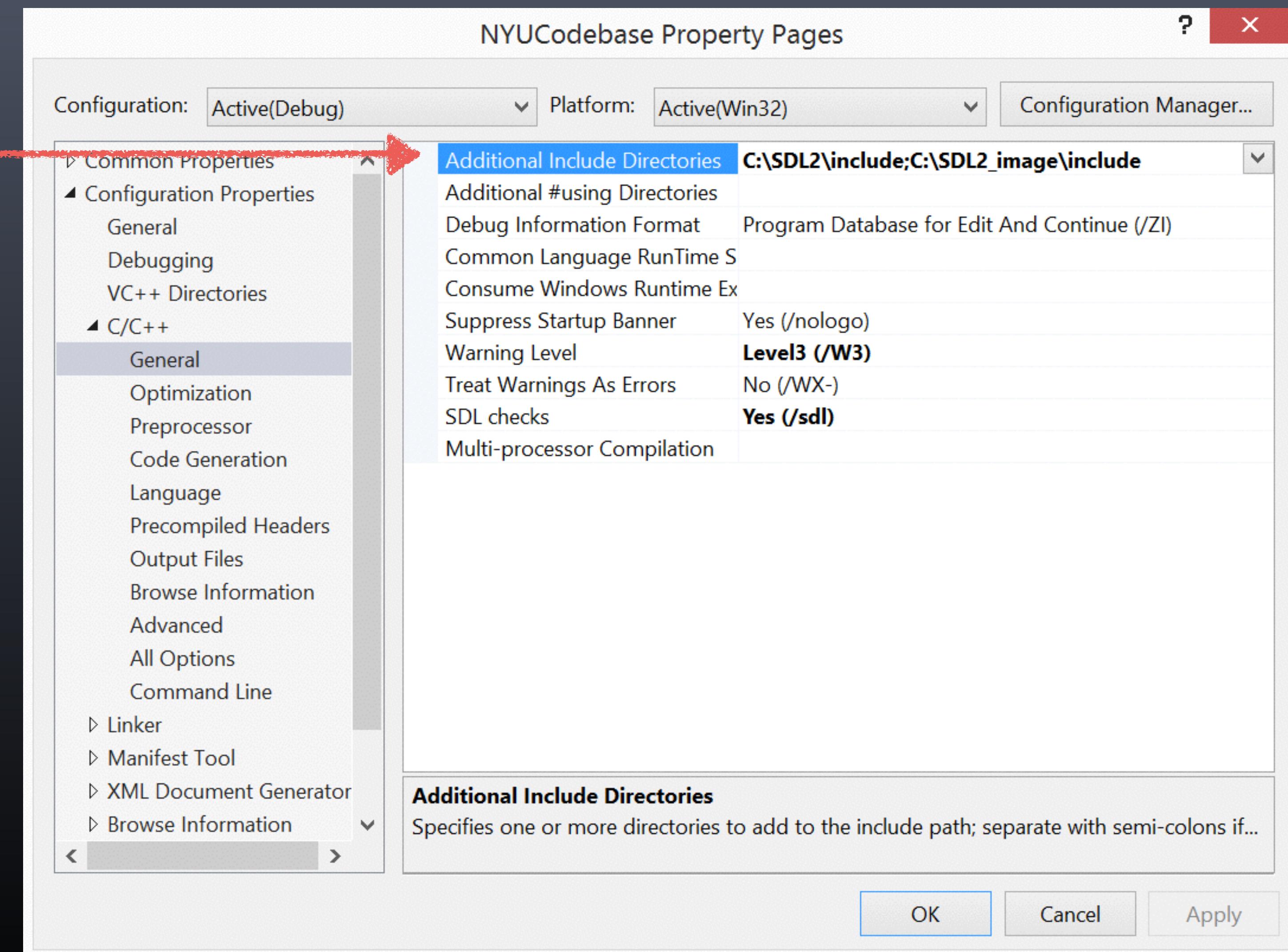


# RIGHT CLICK PROJECT

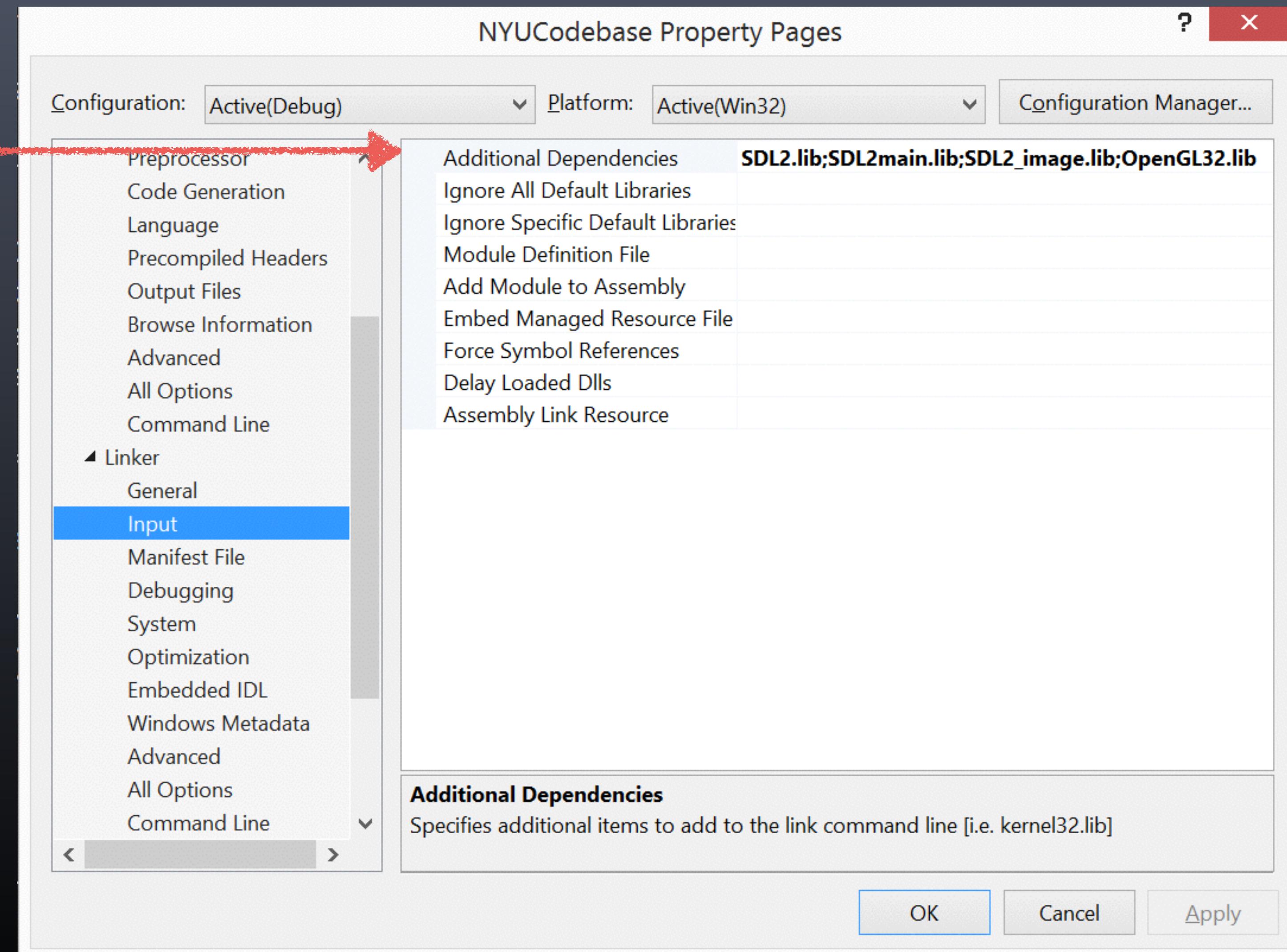


GO TO PROPERTIES

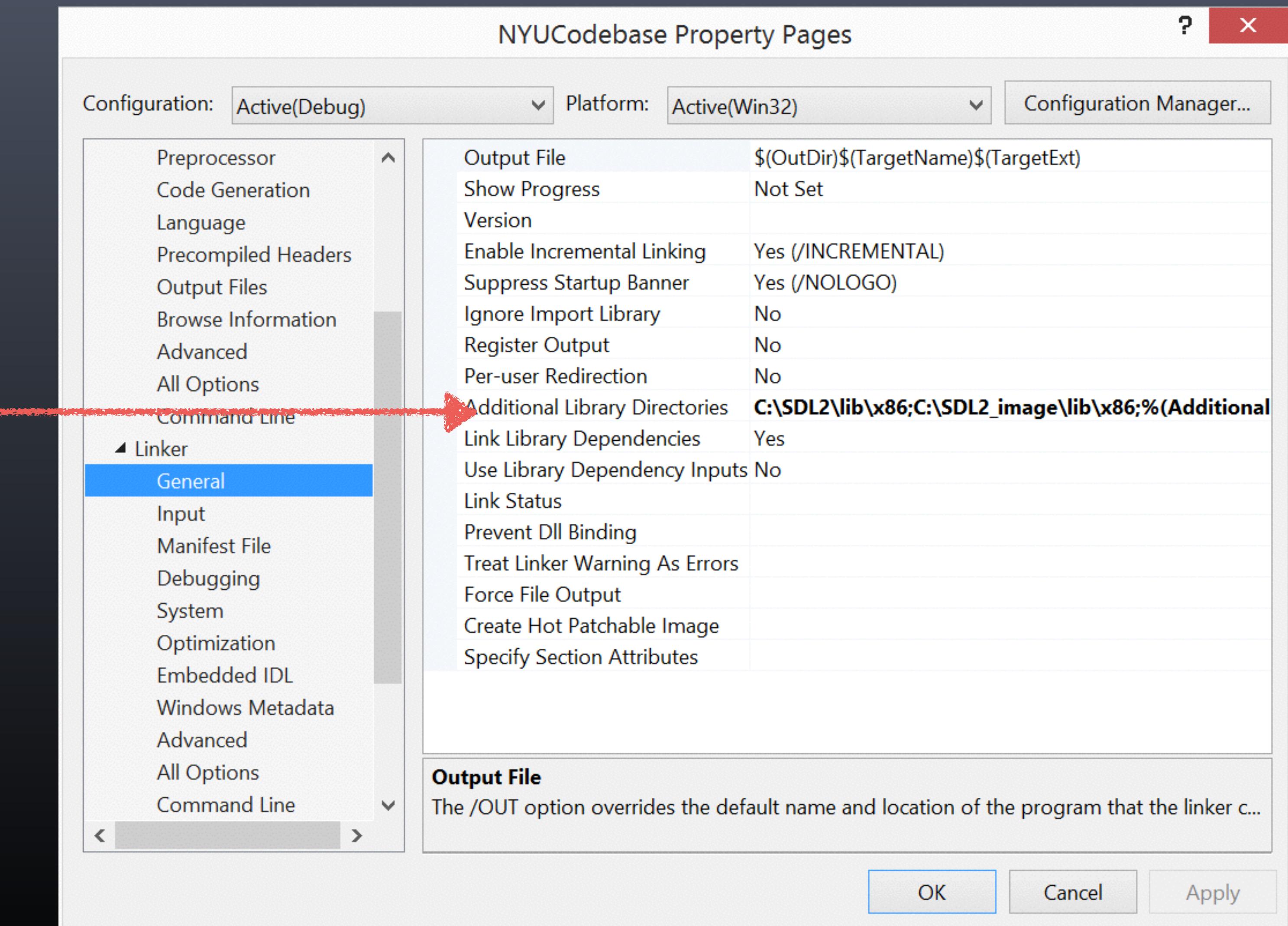
# Add C:\SDL\_mixer\include to additional library directories



# ADD SDL2\_mixer.lib to additional dependencies



# Add C:\SDL\_mixer\lib\x64 to additional library directories



**Do the same thing for both Release and Debug configurations.**

**Copy all DLL files from C:\SDL\_mixer\lib\x86 to where all the other DLL files are in your project.**

# Initializing SDL\_mixer

Include SDL\_mixer header.

```
#include <SDL_mixer.h>
```

```
int Mix_OpenAudio( int frequency, Uint16 format, int channels,  
int chunkszie);
```

Initializes SDL\_mixer with frequency, format, channel and buffer size.

```
Mix_OpenAudio( 44100, MIX_DEFAULT_FORMAT, 2, 4096 );
```

Loading and playing sounds.

# Loading a sound.

```
Mix_Chunk *someSound;  
someSound = Mix_LoadWAV("some_sound.wav");
```

# Playing a sound.

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

Plays a sound on specified channel. You can pass -1 for “channel” to use the first available channel. Loops can be -1 to loop forever.

```
Mix_PlayChannel( -1, someSound, 0);
```

Loading and playing music.

# Loading music.

```
Mix_Music *music;  
music = Mix_LoadMUS( "music.mp3" );
```

# Playing music.

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

Plays specified music. Loops can be -1 to loop forever.

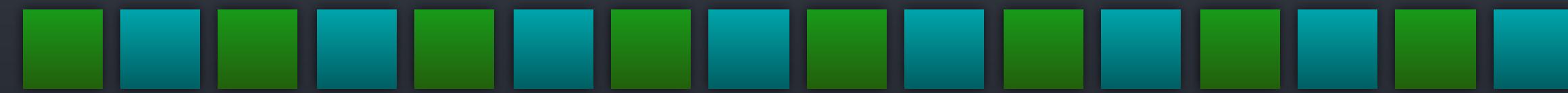
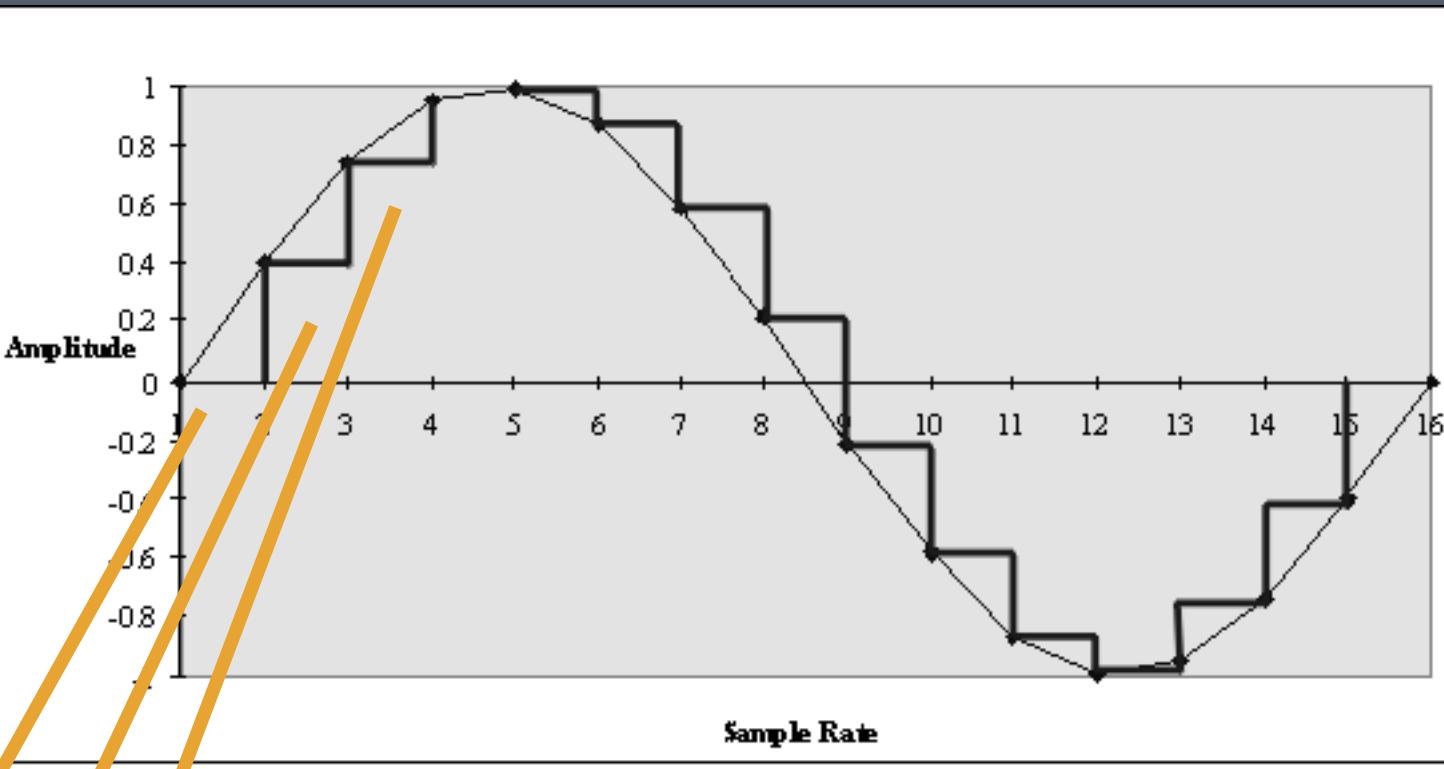
```
Mix_PlayMusic(music, -1);
```

Cleaning up.

Need to clean up music and sounds on quit.

```
DemoApp::~DemoApp() {  
    Mix_FreeChunk(someSound);  
    Mix_FreeMusic(music);  
  
    SDL_Quit();  
}
```

Pure audio buffer (the hard way).



Audio Buffer



Operating system audio mixer

Need to add `SDL_INIT_AUDIO` to `SDL_Init` flags.

```
SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO);
```

# Define your audio callback function.

```
void myAudioCallback(void *userdata, Uint8 *stream, int len) {  
}
```

Open an audio device with your callback and desired settings.

```
SDL_AudioSpec deviceSpec;  
deviceSpec.freq = 44100; // sampling rate (samples a second)  
deviceSpec.format = AUDIO_F32; // audio format  
deviceSpec.channels = 1; // how many channels (1 = mono, 2 = stereo)  
deviceSpec.samples = 512; // audio buffer size in samples (power of 2)  
deviceSpec.callback = myAudioCallback; // our callback function  
  
// open new audio device with our requested settings  
SDL_AudioDeviceID dev = SDL_OpenAudioDevice(NULL, 0, &deviceSpec, 0,  
SDL_AUDIO_ALLOW_FORMAT_CHANGE);  
SDL_PauseAudioDevice(dev, 0); // start playback on this device
```

# The audio callback function will be called by SDL every time it needs us to refill the audio buffer.

Here is an example of an audio callback function that's generating a 440Hz tone at 48000 sample frequency in a one channel float buffer.

```
unsigned int numSamples = 0;

float getAudioForTime(long numSamples) {
    double elapsed = ((double)numSamples)/44100.0;
    return sin(elapsed * 2.0 * M_PI * 440.0);
}

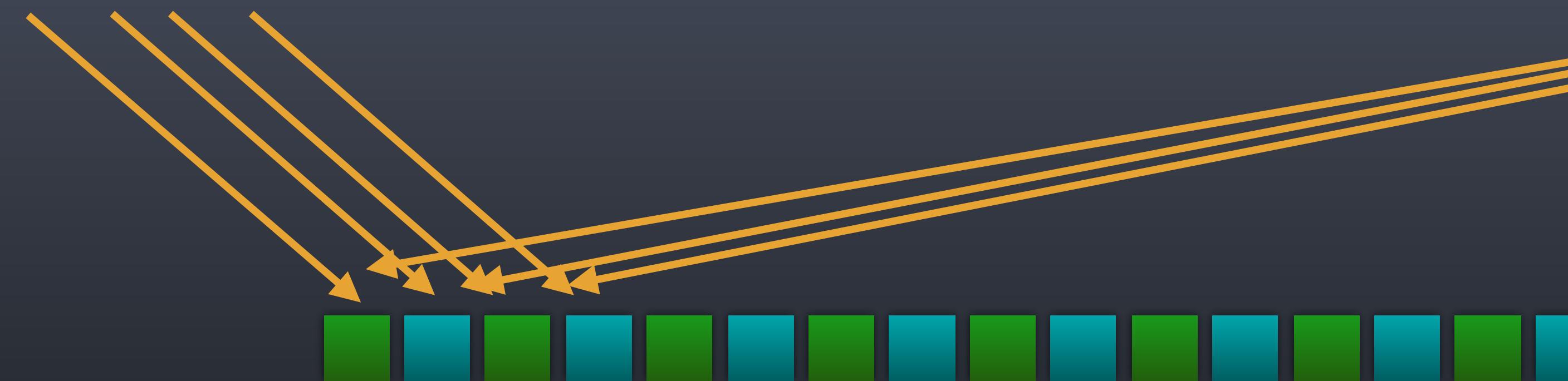
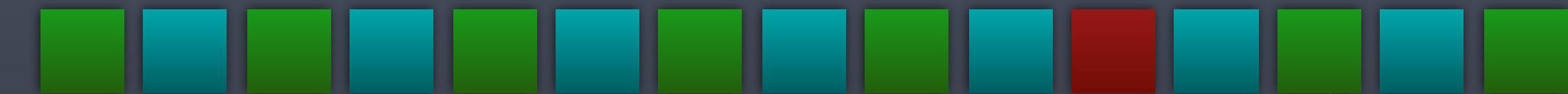
void myAudioCallback(void *userdata, Uint8 *stream, int len) {
    for(int i=0; i < len/4; i++) {
        ((float*)stream)[i] = getAudioForTime(numSamples);
        numSamples++;
    }
}
```

Writing a simple mixer.

**Sound #1**



**Sound #2**



**Audio Buffer**

**Operating system audio mixer**

```
class MixerSound {  
public:  
    Uint32 offset;  
    Uint32 length;  
    Uint8 *buffer;  
    float volume;  
    SDL_AudioFormat format;  
    bool loaded;  
    bool playing;  
    bool loop;  
};  
  
class DemoApp {  
public:  
    std::vector<MixerSound> mixSounds;  
};
```

Create a class for storing loaded sound buffers and some basic sound information. Our app will keep these in a vector.

```
int DemoApp::loadSound(const char *soundFile) {  
  
    Uint8 *buffer;  
    SDL_AudioSpec spec;  
    Uint32 bufferSize;  
  
    if(SDL_LoadWAV(soundFile, &spec, &buffer, &bufferSize) == NULL) {  
        return -1;  
    }  
  
    SDL_AudioCVT cvt;  
    SDL_BuildAudioCVT(&cvt, spec.format, spec.channels, spec.freq,  
deviceSpec.format, deviceSpec.channels, deviceSpec.freq);  
    cvt.len = bufferSize;  
    cvt.buf = new Uint8[bufferSize * cvt.len_mult];  
    memcpy(cvt.buf, buffer, bufferSize);  
  
    SDL_ConvertAudio(&cvt);  
    SDL_FreeWAV(buffer);  
  
    MixerSound sound;  
    sound.buffer = cvt.buf;  
    sound.length = cvt.len_cvt;  
    sound.loaded = true;  
    sound.offset = 0;  
    sound.format = deviceSpec.format;  
    sound.volume = 1.0;  
    sound.playing = false;  
    mixerSounds.push_back(sound);  
  
    return mixerSounds.size()-1;  
}
```

SDL\_LoadWAV can only  
load .wav files.

```
someSound = loadSound("some_wave_file.wav");
```

```
void DemoApp::appAudioCallback(void *userdata, Uint8 *stream, int len) {
    DemoApp *app = (ClassDemoApp*) userdata;
    float *mixBuffer = new float[len/4];

    memset(stream, 0, len);
    memset(mixBuffer, 0, len);
    int numSoundsMixed = 0;
    for(int i=0; i < app->mixerSounds.size(); i++) {
        MixerSound *sound = &app->mixerSounds[i];

        if(sound->loaded && sound->playing) {
            for(int s=0; s < len/4; s++) {
                float *sourceBuffer = (float*) (sound->buffer+sound->offset);
                mixBuffer[s] += (sourceBuffer[s] * sound->volume);
            }
            numSoundsMixed++;
        }

        sound->offset += len;
        if(sound->offset >= sound->length-len) {
            if(sound->loop) {
                sound->offset = 0;
            } else {
                sound->playing = false;
            }
        }
    }

    if(numSoundsMixed > 0) {
        for(int s=0; s < len/4; s++) {
            ((float*)stream)[s] = mixBuffer[s] / (float)numSoundsMixed;
        }
    }
    free(mixBuffer);
}
```

```
deviceSpec.callback = DemoApp::appAudioCallback;
```

# Playing a sound.

```
void DemoApp::playSound(int soundIndex, bool loop) {  
    if(soundIndex >= 0 && soundIndex < mixerSounds.size()) {  
        mixerSounds[soundIndex].playing = true;  
        mixerSounds[soundIndex].offset = 0;  
        mixerSounds[soundIndex].loop = loop;  
    }  
}
```

```
playSound(someSound, false);
```

Sound resources.

# SFXR

<http://www.superflashbros.net/as3sfxr/>

The screenshot shows the SFXR interface, a web-based tool for generating sounds. On the left, a vertical menu lists various sound generator types: GENERATOR (PICKUP/COIN, LASER/SHOOT, EXPLOSION, POWERUP, HIT/HURT, JUMP, BLIP/SELECT), MUTATE, RANDOMIZE, BACK, FORWARD, and SFGBTOM. The central area is titled "MANUAL SETTINGS" and contains four tabs: SQUAREWAVE, SAWTOOTH, SINEWAVE, and NOISE. Below these tabs is a large grid of sliders and labels. The labels include: ATTACK TIME, SUSTAIN TIME, SUSTAIN PUNCH, DECAY TIME, START FREQUENCY, MIN FREQUENCY, SLIDE, DELTA SLIDE, VIBRATO DEPTH, VIBRATO SPEED, CHANGE AMOUNT, CHANGE SPEED, SQUARE DUTY, DUTY SWEEP, REPEAT SPEED, Phaser OFFSET, Phaser SWEEP, LP FILTER CUTOFF, LP FILTER CUTOFF SWEEP, LP FILTER RESONANCE, HP FILTER CUTOFF, HP FILTER CUTOFF SWEEP, and PLAY ON CHANGE. To the right of the grid, there are instructions: "CLICK ON LABELS TO RESET SLIDERS", "COPY/PASTE SETTINGS TO SHARE SOUNDS", and "BASED ON SFXR BY TOMAS PETTERSSON". At the bottom right are buttons for VOLUME, PLAY SOUND, LOAD SOUND, SAVE SOUND, EXPORT .WAV, 44100 Hz, and 16-BIT.

# FREE MUSIC ARCHIVE

<http://freemusicarchive.org/>



Assignment.

- Make a simple scrolling sidescroller or topdown game demo.
- It must use a tilemap.
- It must scroll.
- It must be either procedurally generated or load levels from a file.
- Must have music and at least two sound effects.
- You have two weeks.