Theoretical Backgrounds of Audio & Graphics

Oscillators & Sound Generation

Till Bovermann | Dr.-Ing.
Audio & Interactive Media Technologies

Filmuniversität Babelsberg
KONRAD WOLF

Winter term 2019/2020

Top LOTs

1. mathematical representation of a simple single-frequency sound wave
2. calculate a sine wave at a certain frequency & for a certain period of time
3. notion of an audio buffer & how to generate sound

Recap

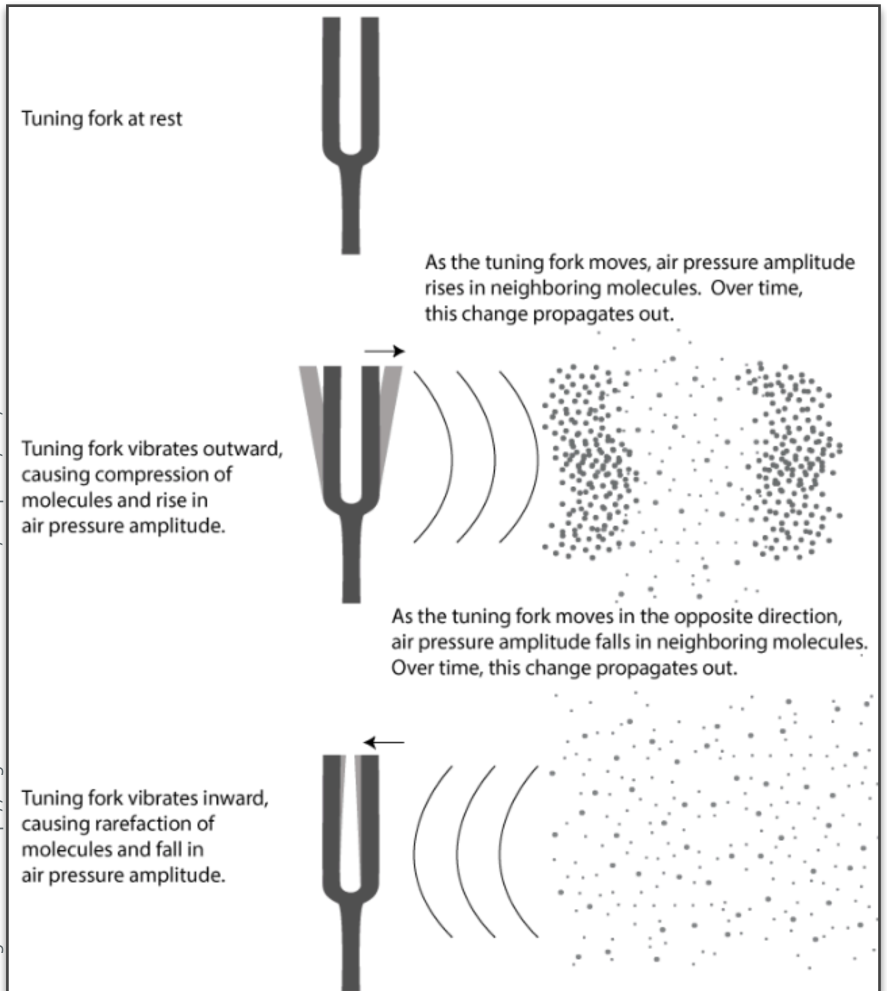


Image credit: http://digitalsoundandmusic.com/chapters/ch2/

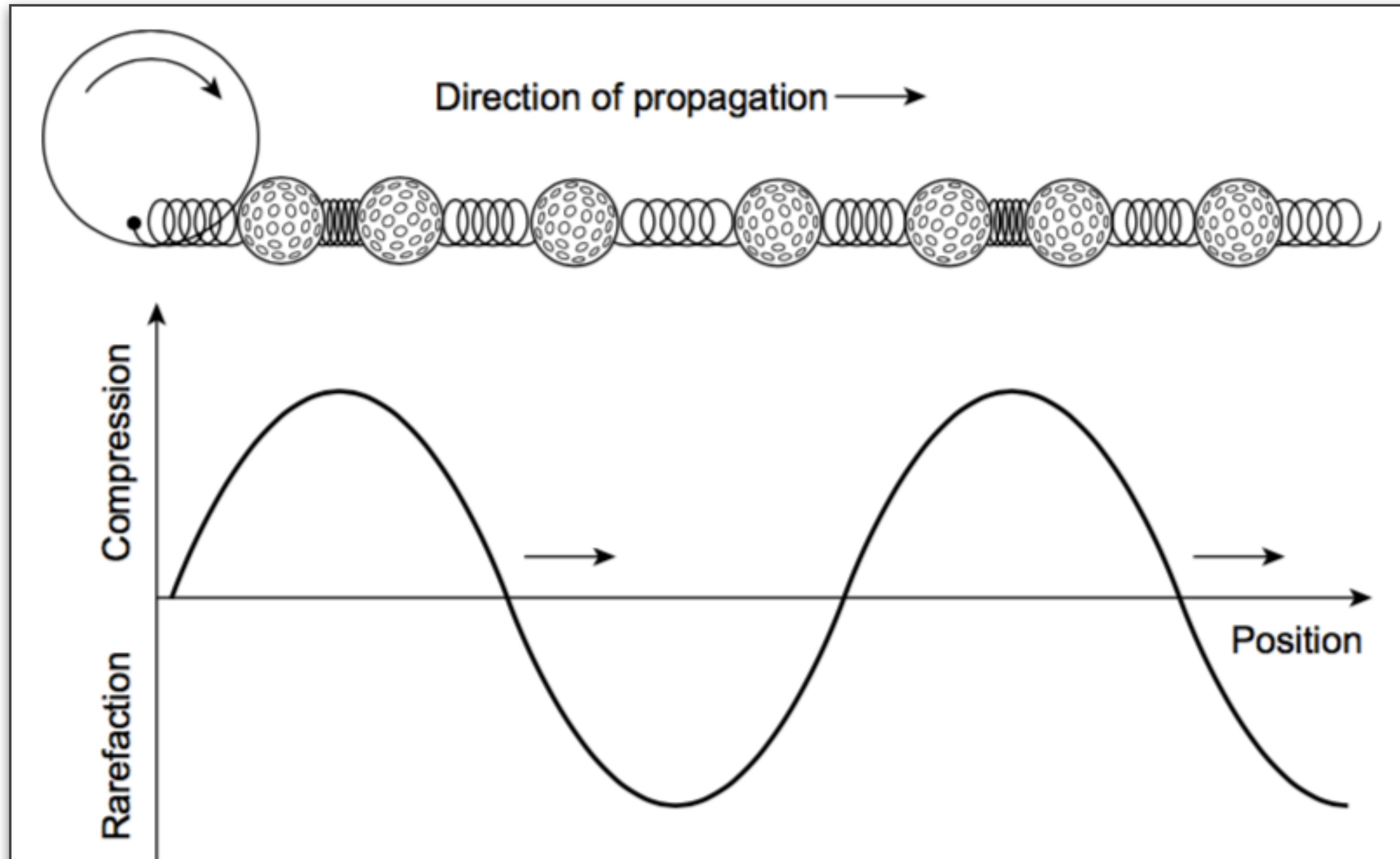


Image credit Howard, D.M. & Angus, J.A.S. (2009): Acoustics and Psychoacoustics. 4th Edition. Oxford, UK: Focal Press.

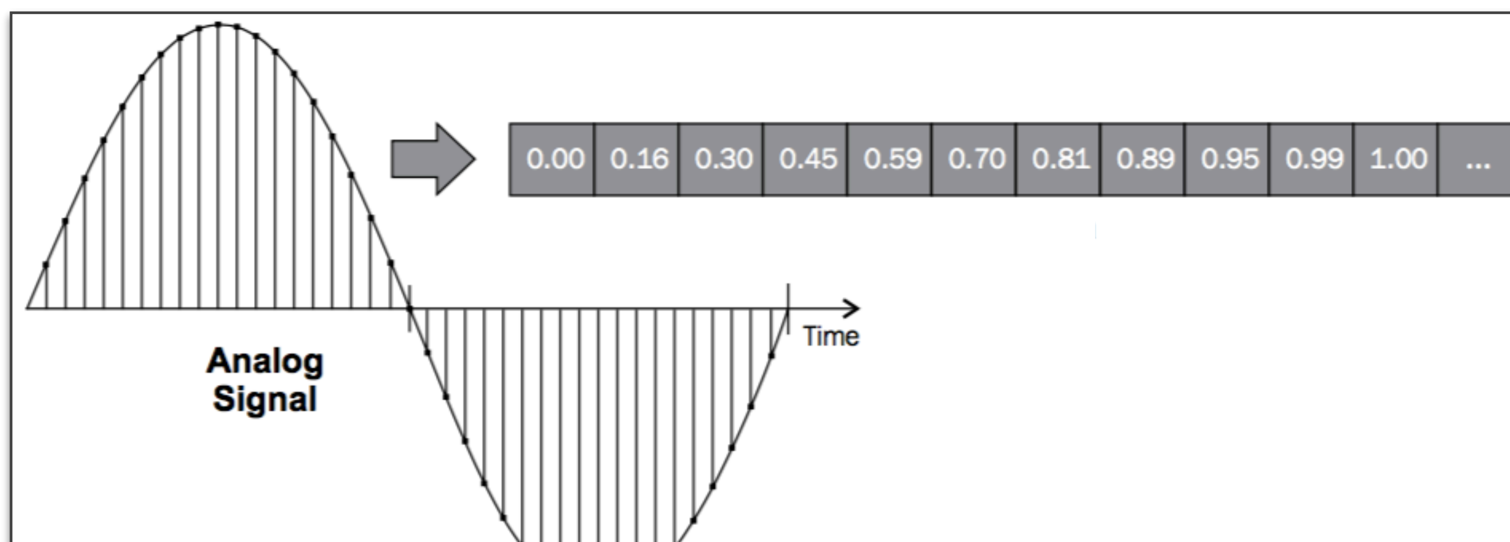
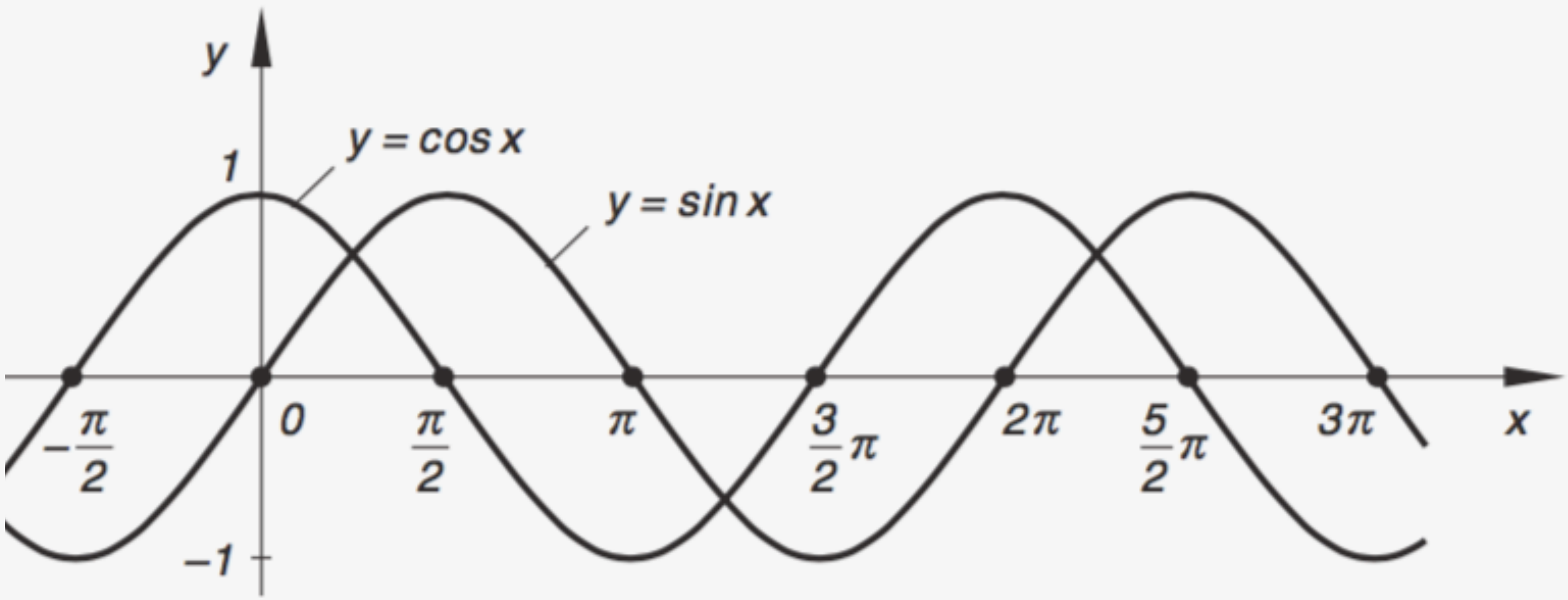


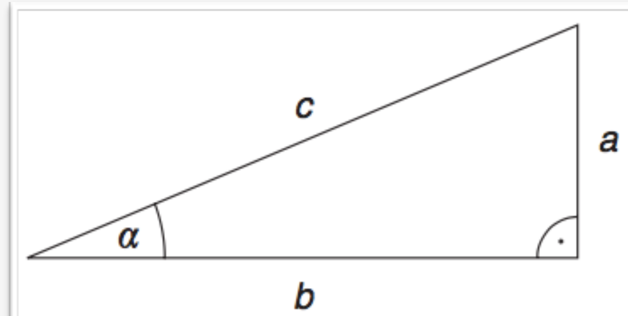
Image credit: Gouveia, D. (2013): Getting Started with C++ Audio Programming for Game Development. Birmingham, UK: Packt Publishing Ltd.

Trigonometric Functions

- Trigonometric functions **sine** & **cosine** describe oscillations well
- They have a **period of 2π (omega)** and are phased shifted by π/2



common
schoolbook definition



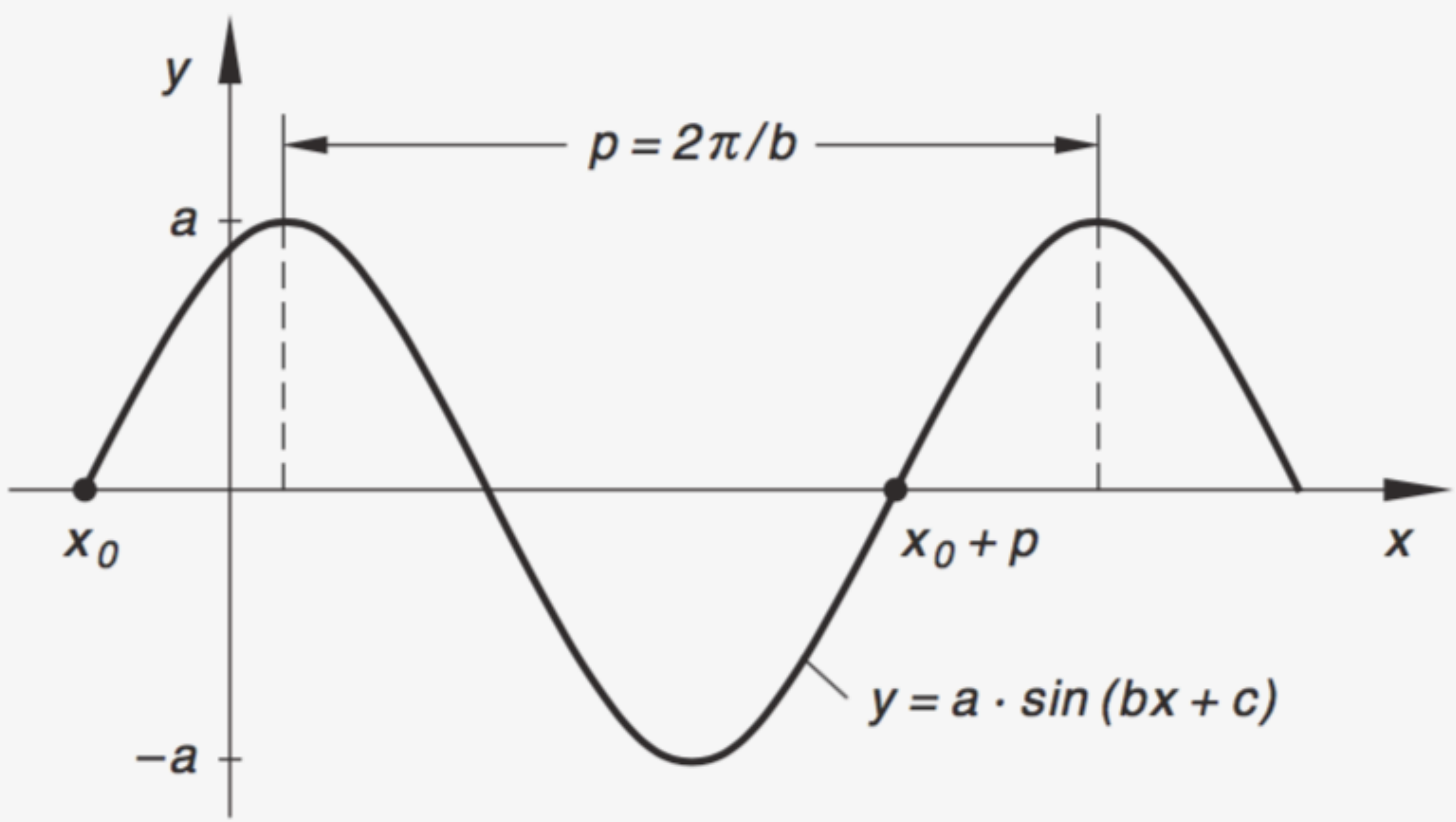
$$\sin\alpha = \frac{\text{Gegenkathete}}{\text{Hypotenuse}} = \frac{a}{c}$$

$$\cos\alpha = \frac{\text{Ankathete}}{\text{Hypotenuse}} = \frac{b}{c}$$

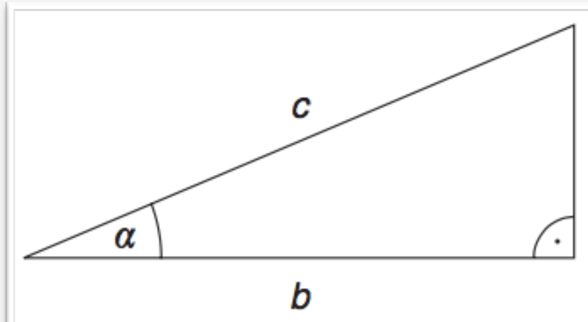
Images credit: Papula, Lothar (2014): **Mathematik für Ingenieure & Naturwissenschaftler Band 1.** 14. überarb. Aufl., Wiesbaden: Springer Vieweg.

Trigonometric Functions

- The general mathematical equation of a sine function ***y = sin(x)*** is given by ***y = a * sin (b*x + c)***



common
schoolbook definition



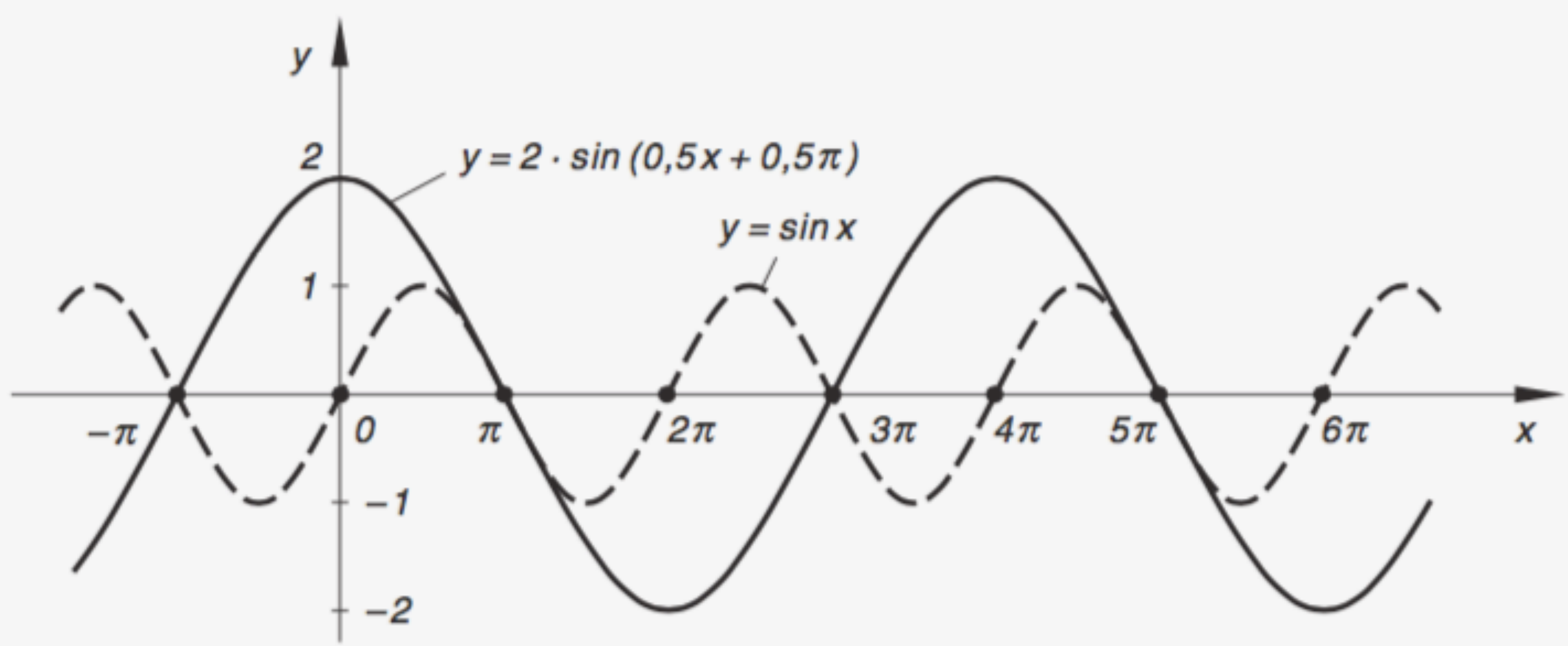
$$\sin \alpha = \frac{\text{Gegenkathete}}{\text{Hypotenuse}} = \frac{a}{c}$$

$$\cos \alpha = \frac{\text{Ankathete}}{\text{Hypotenuse}} = \frac{b}{c}$$

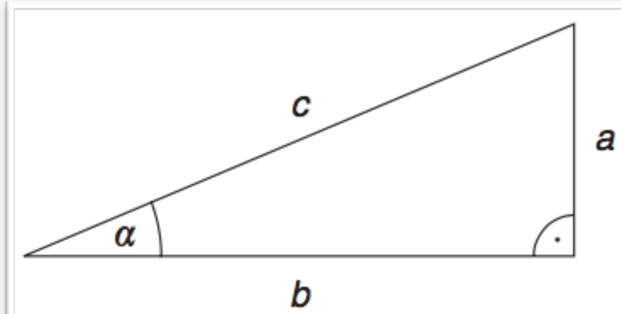
Images credit: Papula, Lothar (2014): **Mathematik für Ingenieure & Naturwissenschaftler Band 1.** 14. überarb. Aufl., Wiesbaden: Springer Vieweg.

Trigonometric Functions

- With the general sine wave equation we can represent any kind of single-frequency wave



common
schoolbook definition



$$\sin\alpha = \frac{\text{Gegenkathete}}{\text{Hypotenuse}} = \frac{a}{c}$$

$$\cos\alpha = \frac{\text{Ankathete}}{\text{Hypotenuse}} = \frac{b}{c}$$

Images credit: Papula, Lothar (2014): **Mathematik für Ingenieure & Naturwissenschaftler Band 1.** 14. überarb. Aufl., Wiesbaden: Springer Vieweg.

Geometric Representation

- In geometric terms, sine and cosine can be expressed as a point **p = (cos(a), sine(a))** that moves around the unit circle

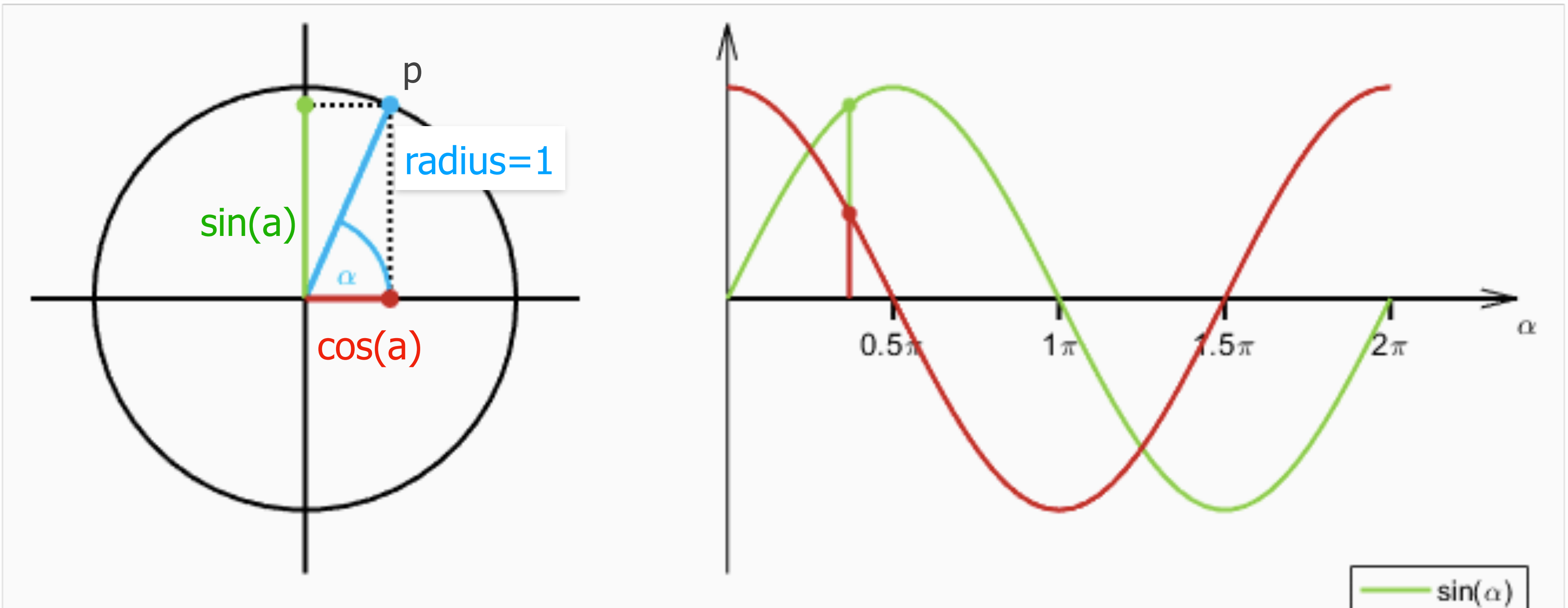


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Geometric Representation

- Point p is often referred to as **phasor** as it represents the phase shift of the equation at time t = 0

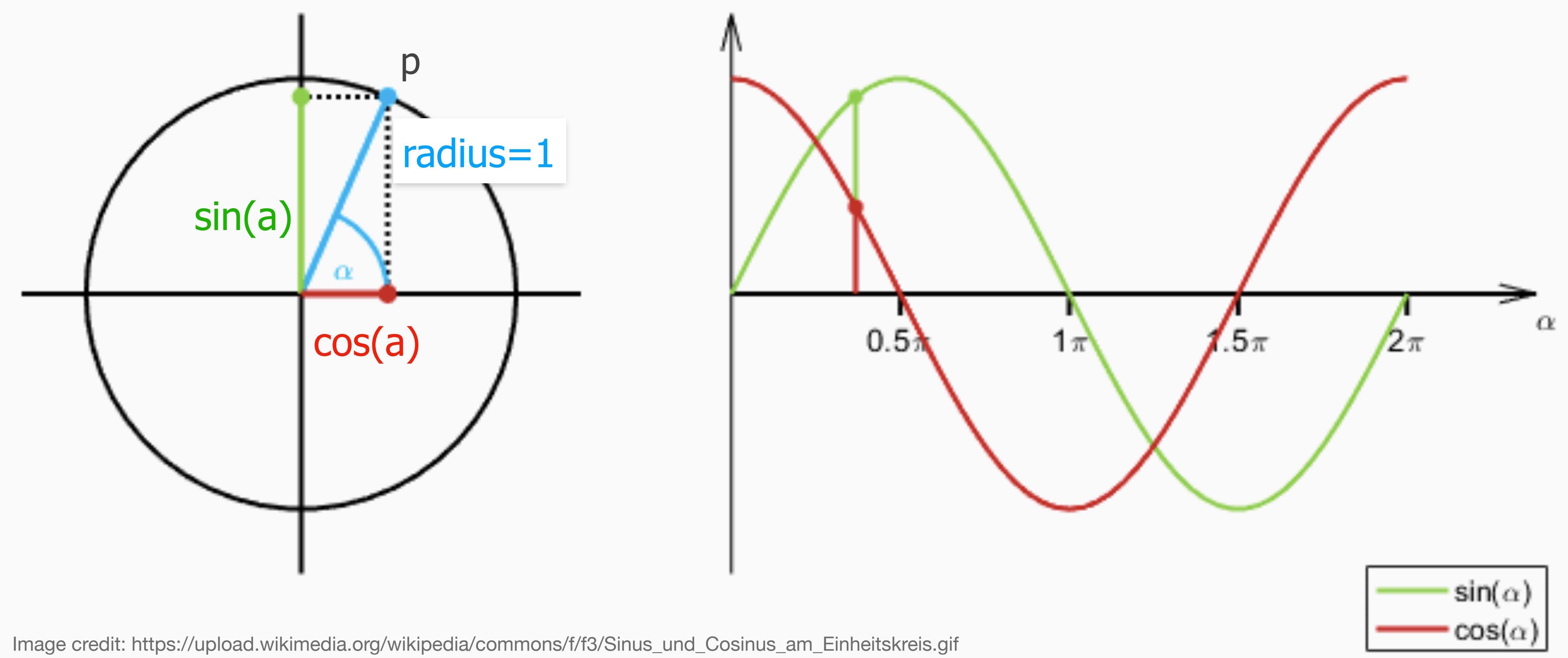


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Oscillators

- A continuous process that grows the angle and lets the **phasor** move around the circle is called an **oscillator**

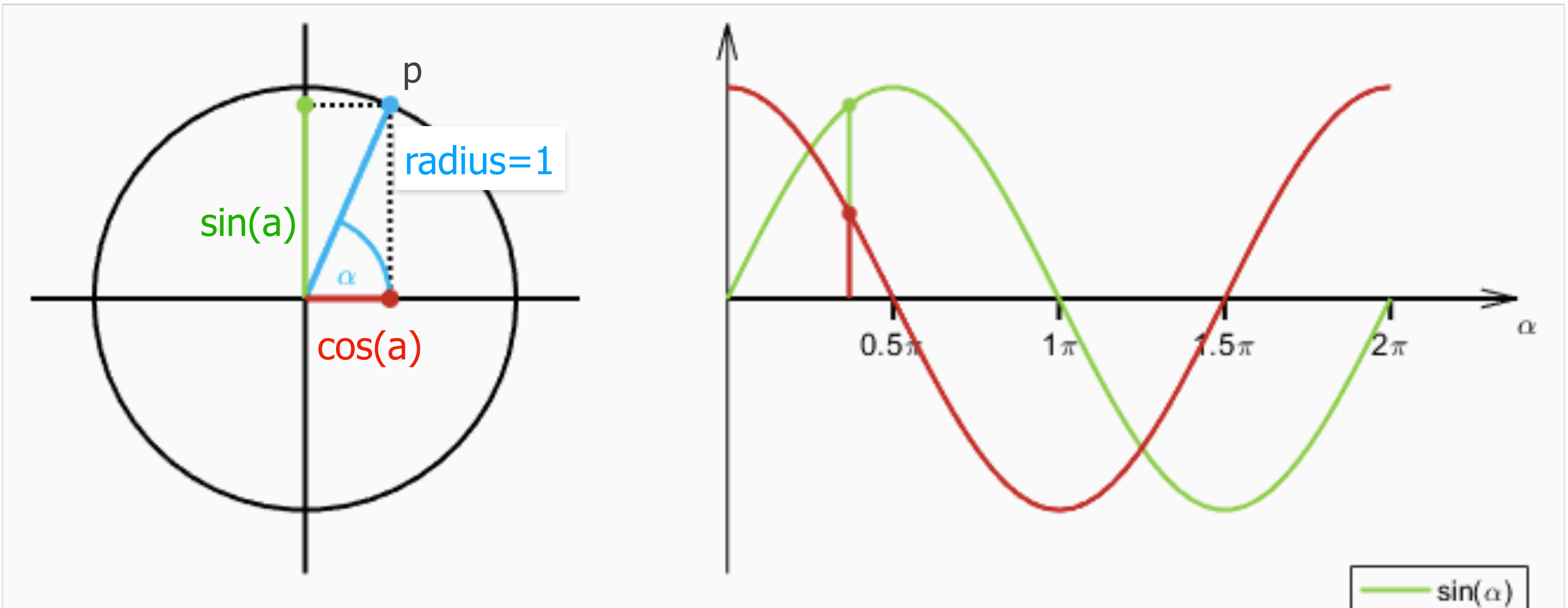


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Oscillators

- An oscillator generates a periodic signal, i.e., a sine wave and relates it to the notion of time — p moves around the circle in a certain amount of time / at a certain frequency

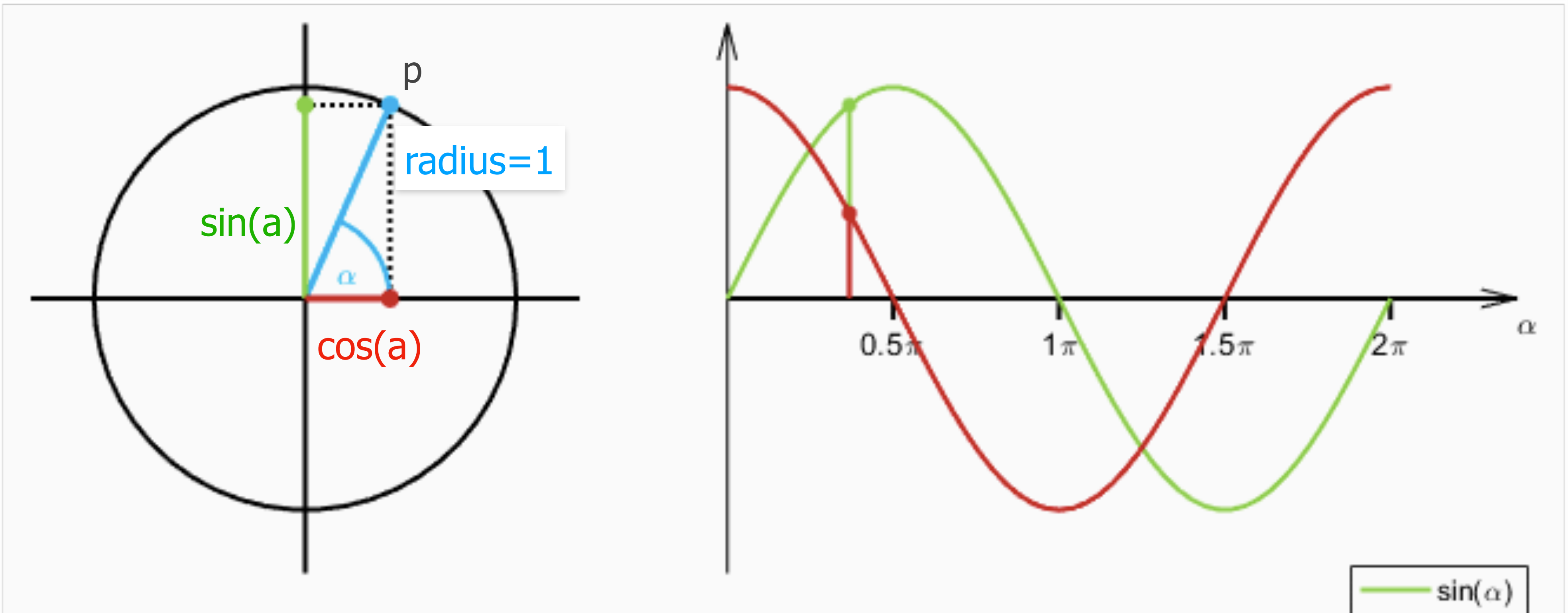


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Oscillators

- An **audio oscillator** produces 16Hz to 20kHz frequencies
  - A low-frequency-oscillator (LFO) produces frequencies < 20 Hz
  - LFOs are used in electronic music, e.g. to create timbral, frequency, or amplitude variations

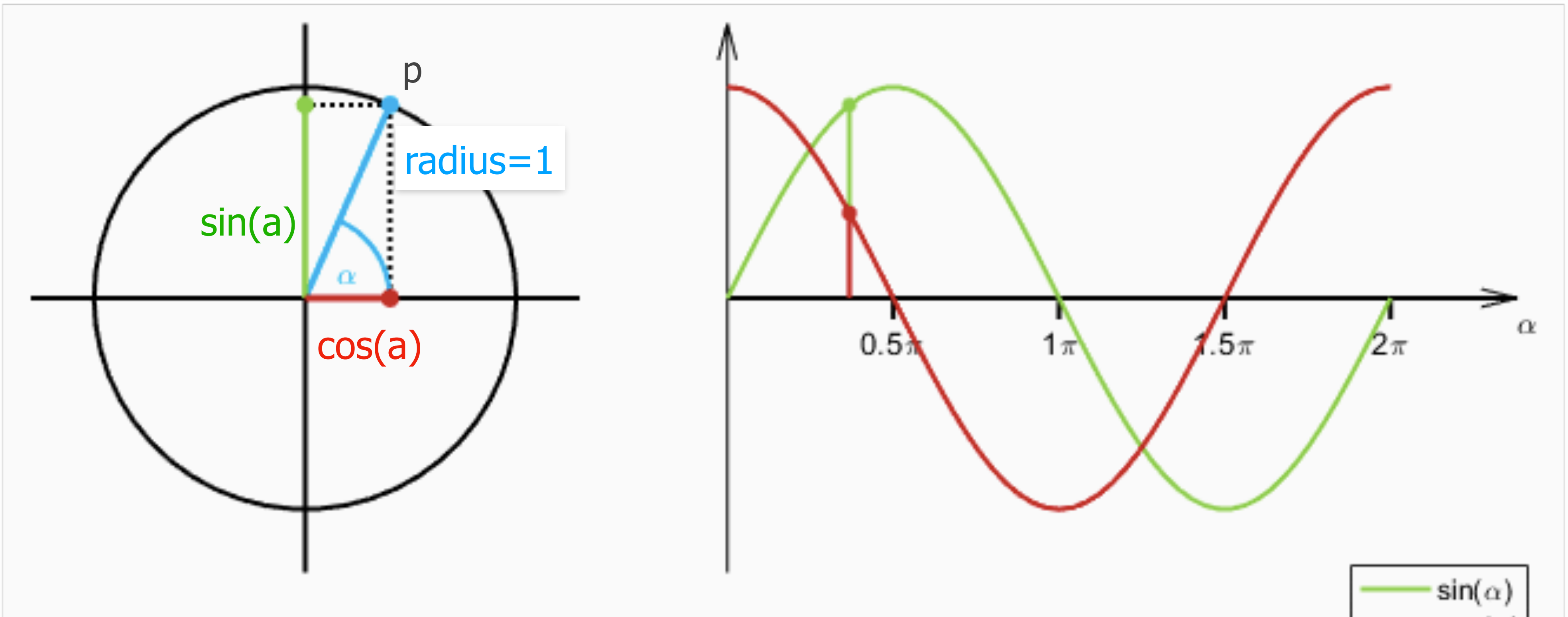


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Sound & Sine Waves

- *How can we describe such an oscillator formally so that we can use an equation to actually generate a sound wave?*

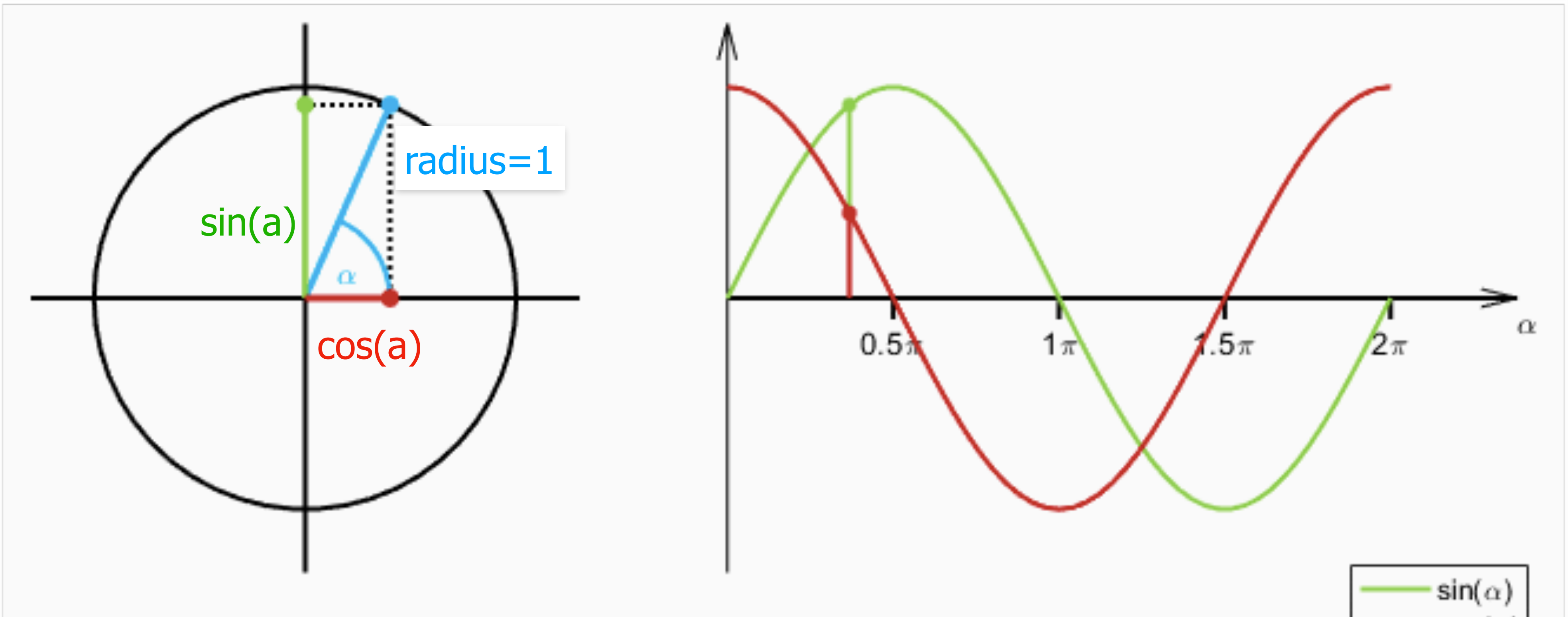


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Sound & Sine Waves

- The general form to create a single-freq sound wave, usually referred to as **sinusoid**, is then expressed by $\boldsymbol{y = A \sin(2\pi f t + ø)}$

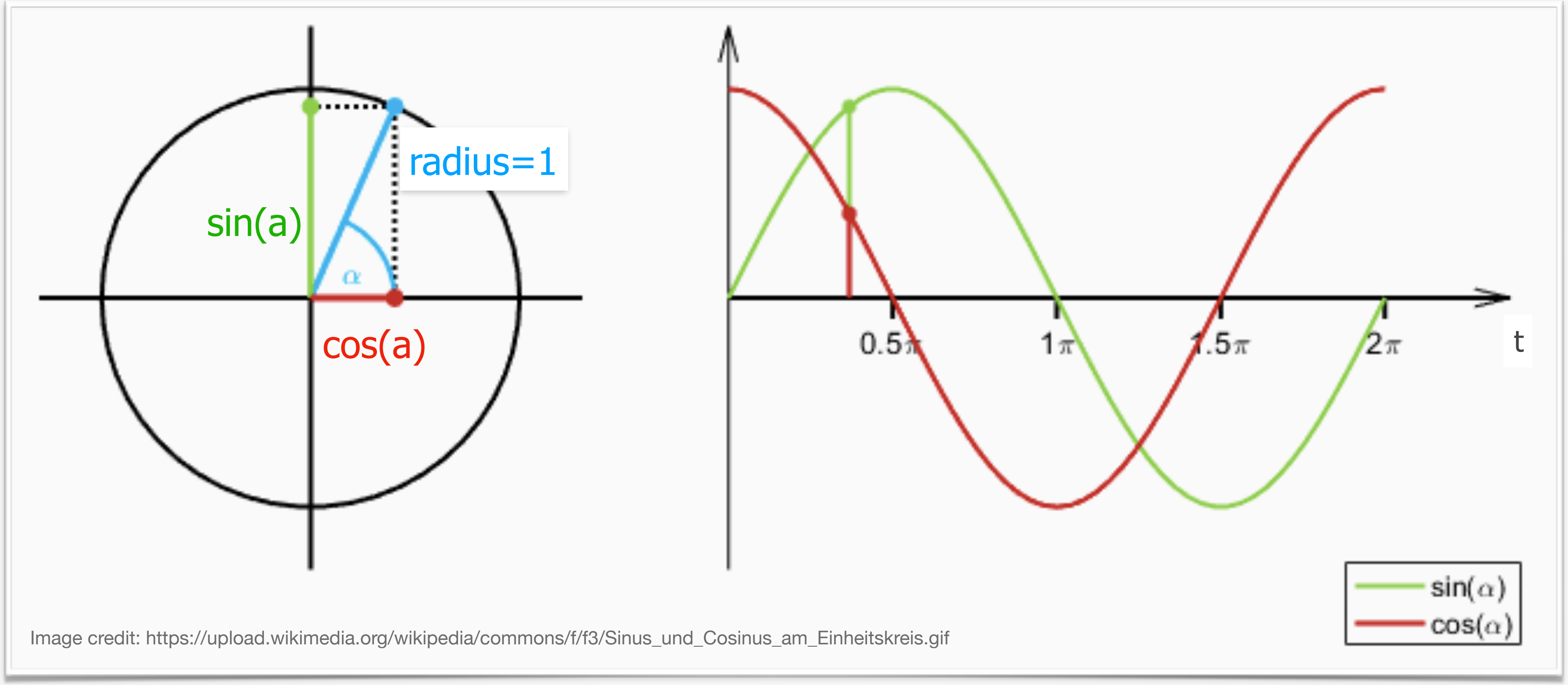


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Sound & Sine Waves

- ***A*** expresses the Amplitude
- **2π*f*** = 2π/T expresses the angular frequency *omega*
- **ø** expresses the phase shift, i.e., where the sinusoid starts at t=0
- **t** expresses the variable of time in seconds / time index

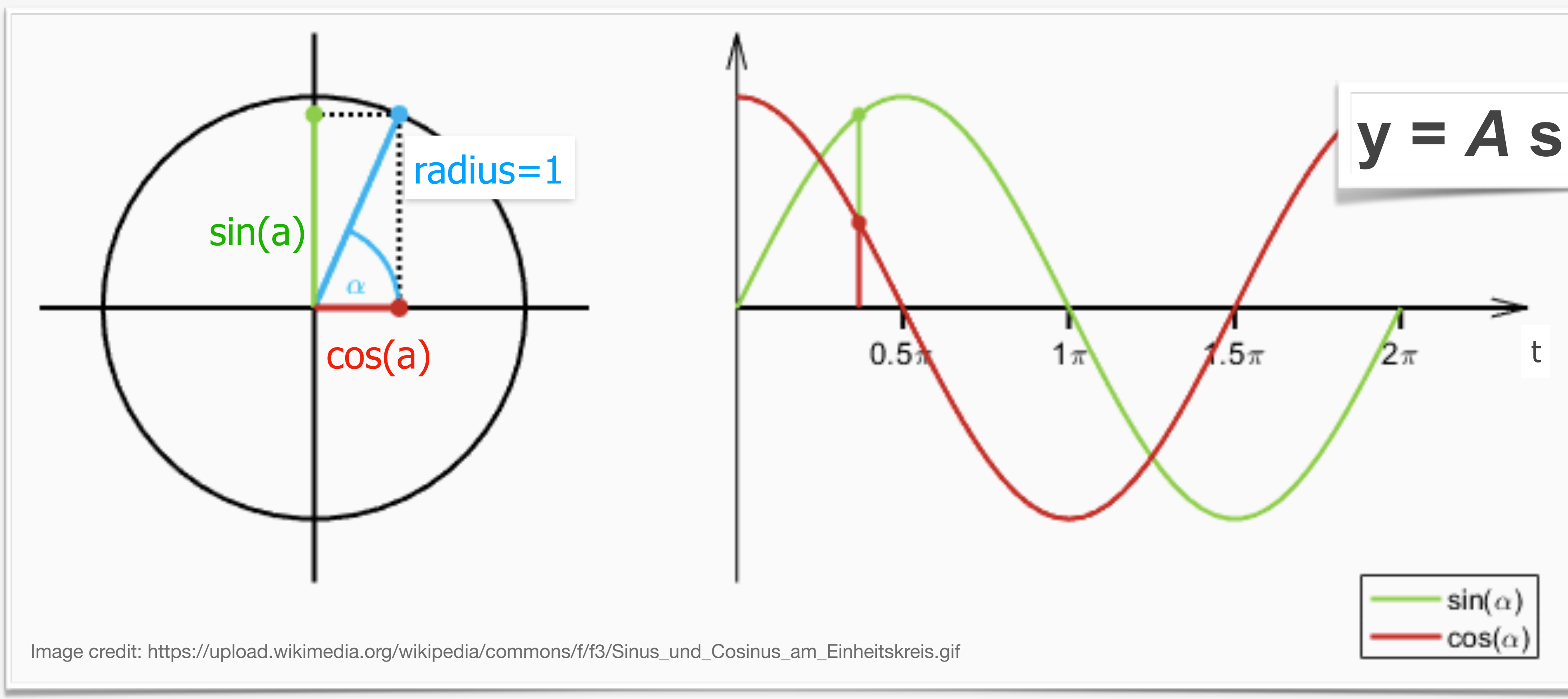
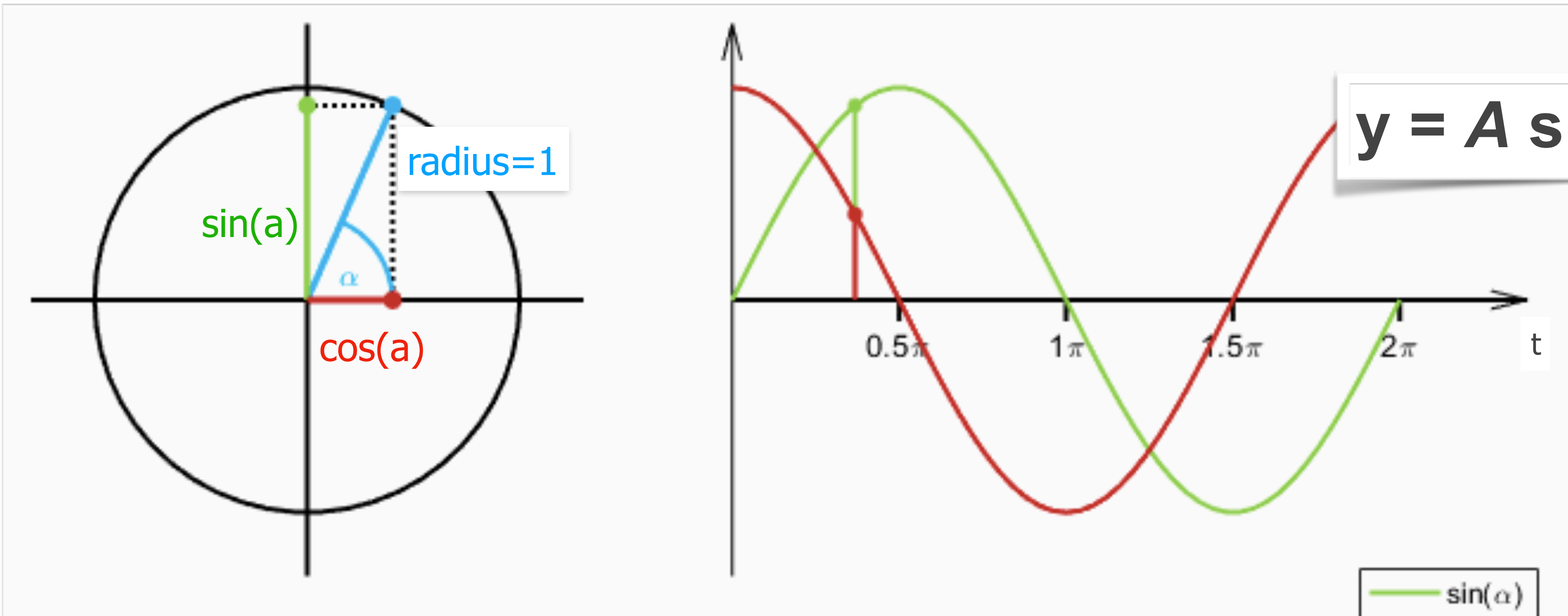


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

y = *A* sin(2π*f*t + ø)

Sound & Sine Waves

- With these information we can now define a periodic signal at any future time t and at any audible frequency 20Hz < f < 20kHz

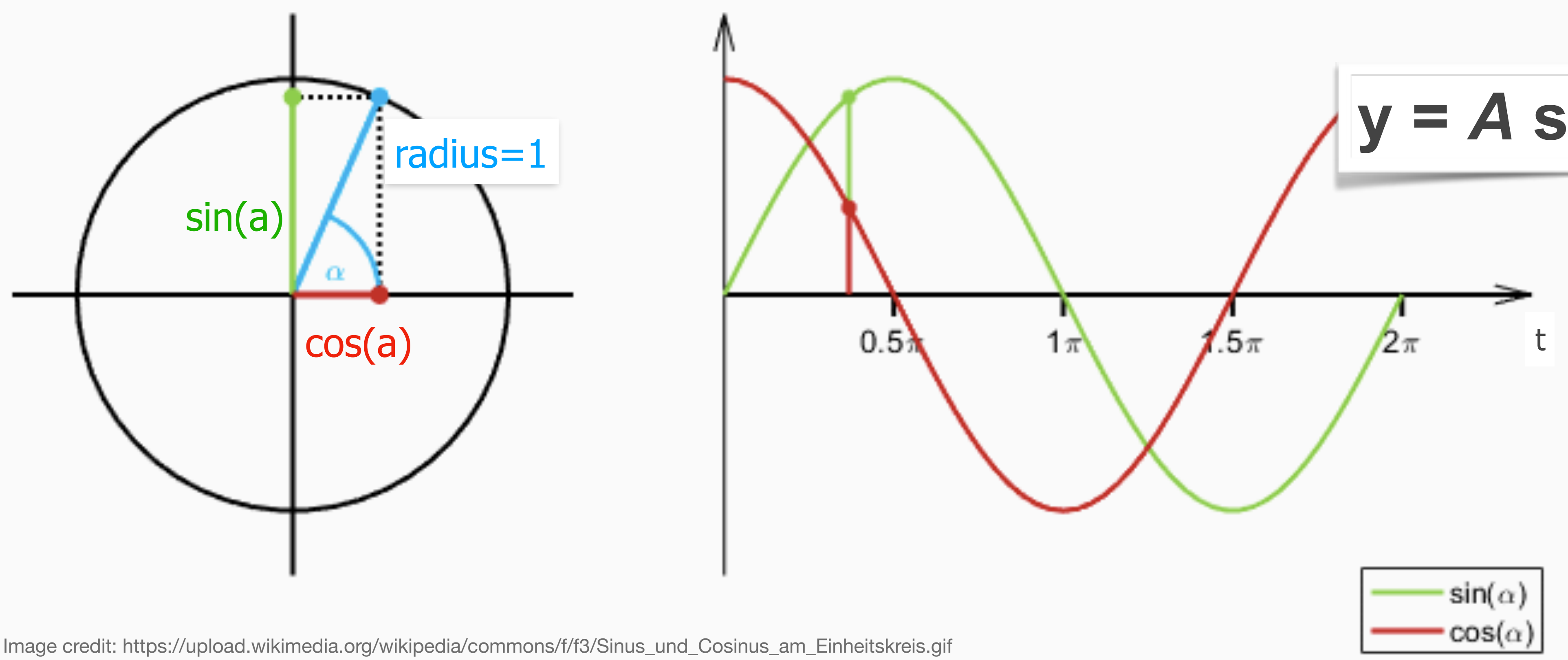


$$y = A \sin(2\pi f t + ø)$$

Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Sound & Sine Waves

- *How do you calculate the 5 seconds of a 5 Hz sound wave, A=1?*
    - ***y = 1 * sin(2π * 5 * t) => for t = 0 to 5 ?***

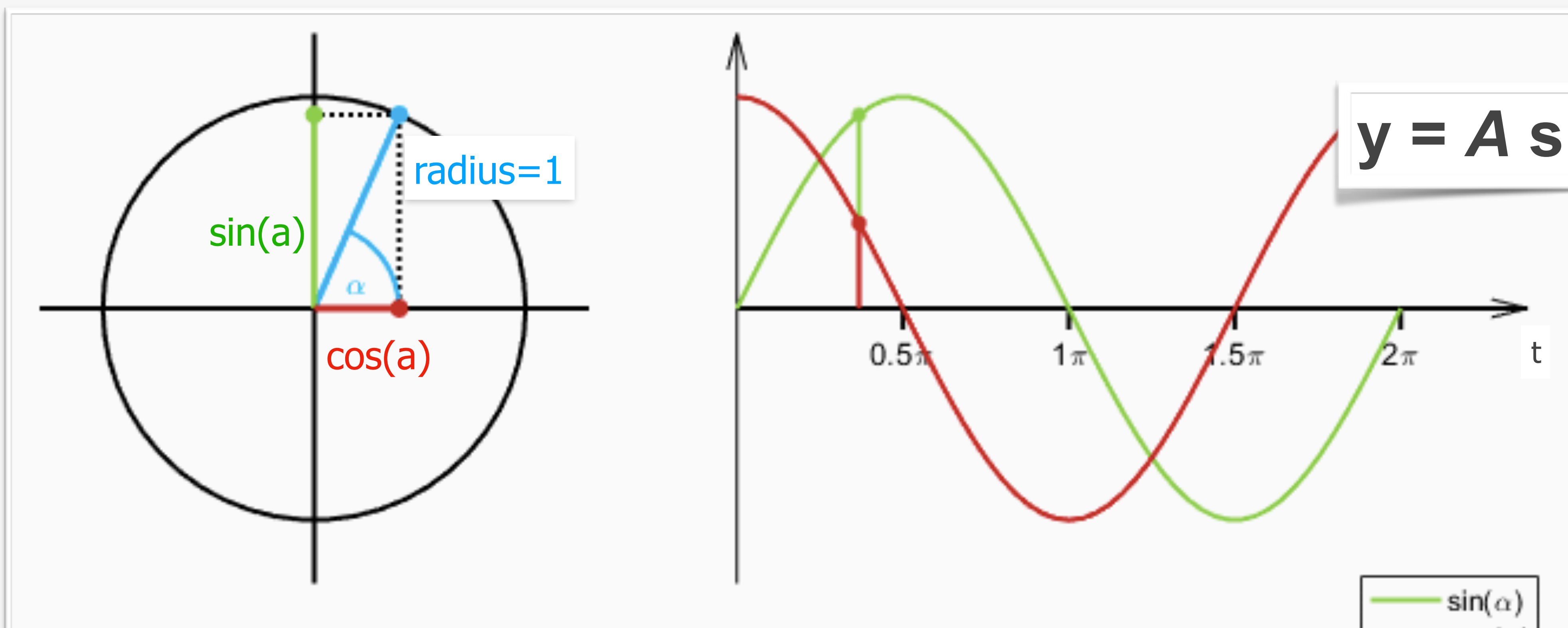


$$y = A \sin(2\pi f t + ø)$$

Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Sound & Sine Waves

- *What about t in this calculation?*
  - ***y = 1 * sin(2π * 5 * t) => for t = 0 to 5 ?***
- *t needs to be related to the sampling rate!*



y = *A* sin(2π*f*t + ø)

Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_und_Cosinus_am_Einheitskreis.gif

Sound & Sine Waves

- Sine waves are periodic functions that serve well to describe single frequency wave forms

- The general form of a sine wave is referred to as sinusoid & allows us to generate any kind of single-frequency sine wave

- When applied to sound wave generation, sinusoid equations can be used to generate the amplitude values of a sound wave for a certain period of time at a specific frequency

Filling the Audio Buffer

- Create a digital sound of **440 Hz** that lasts for **2 seconds** and plays back at a frequency of **44.1 kHz** (sampling rate)

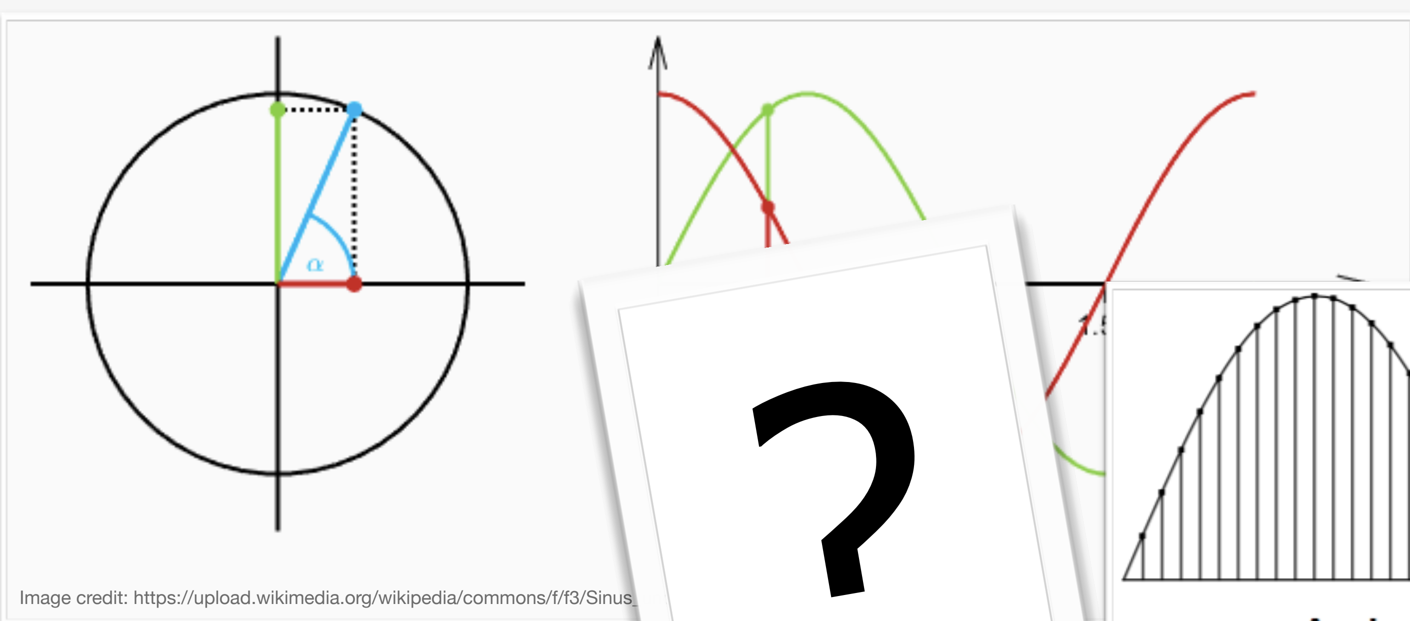


Image credit: https://upload.wikimedia.org/wikipedia/commons/f/f3/Sinus_

$$y = A \sin(2\pi f t + \phi)$$

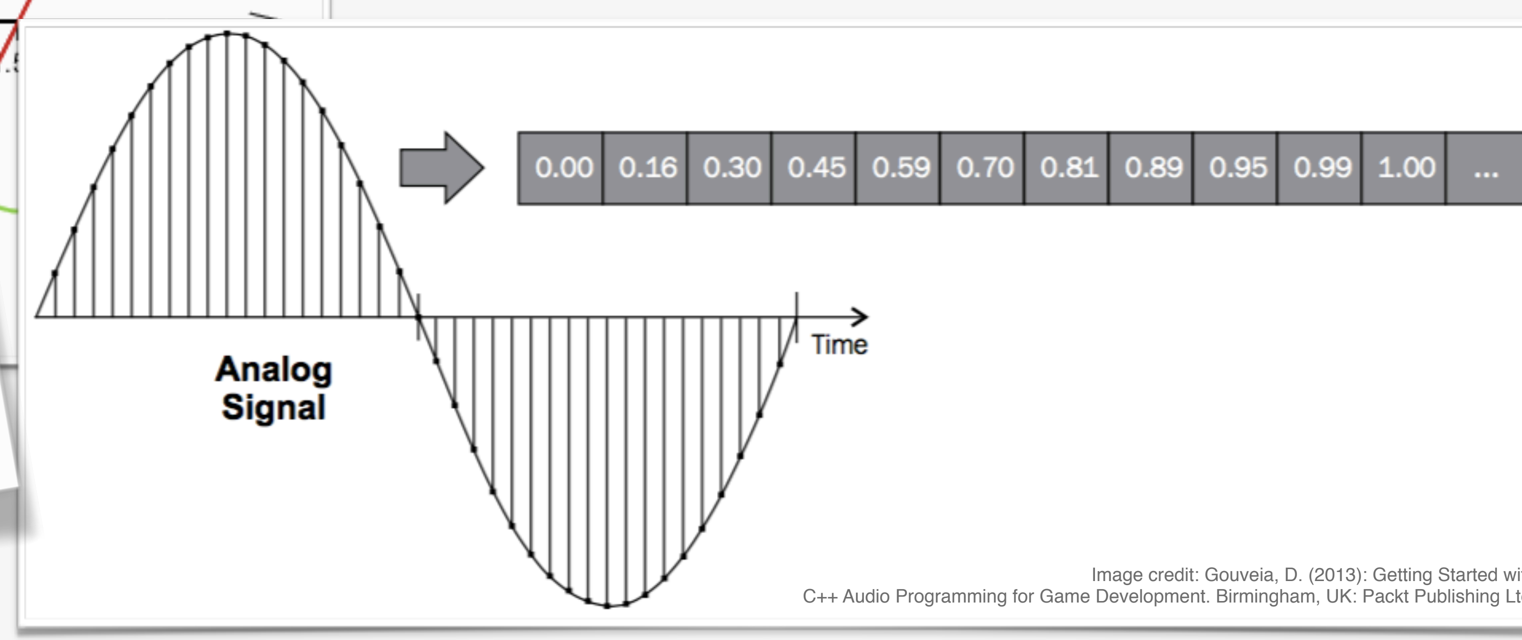


Image credit: Gouveia, D. (2013): Getting Started with C++ Audio Programming for Game Development. Birmingham, UK: Packt Publishing Ltd.

Filling the Audio Buffer

- Create a digital sound of **440 Hz** that lasts for **2 seconds** and plays back at a frequency of **44.1 kHz** (sampling rate)

$$y = A \sin(2\pi f t + \phi)$$

- **Todo**
  - Create an audio buffer of size 2 seconds: **2 * 44100**
  - Fill the audio buffer with sine wave values at frequency 440 Hz
  - Relate time index t to the sampling frequency: **t / 44100**

Filling the Audio Buffer

- Create a digital sound of **440 Hz** that lasts for **2 seconds** and plays back at a frequency of **44.1 kHz** (sampling rate)

$$y = A \sin(2\pi f t + ø)$$

```
audioBuffer = array[44100 * 2];

for (t = 0; t < 88200; t++) {
  A = 1;
  y = A * sin (2π * 440 * (t / 44100));
  audioBuffer[ t ] = y;
}
```

More on Audio Buffers

Continuous Signal

- Audio buffers are data structures used to track all values & properties required to reconstruct the waveform of a continuous analog signal

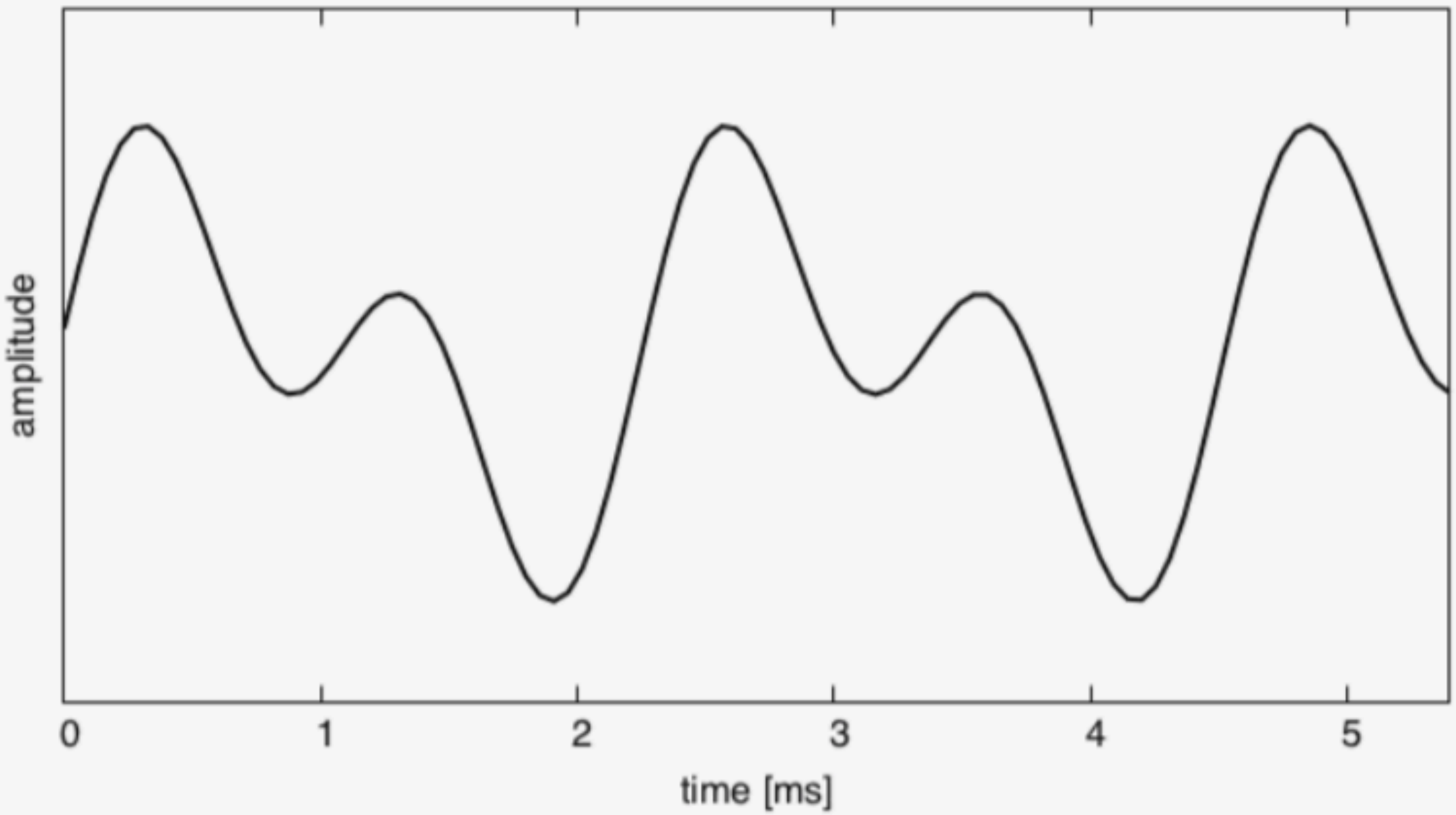


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry**. Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Sampled Signal

- Audio buffers store the sampled amplitude values per time index in a data array
- Additionally, they store the sampling rate for correctly reading & writing the data

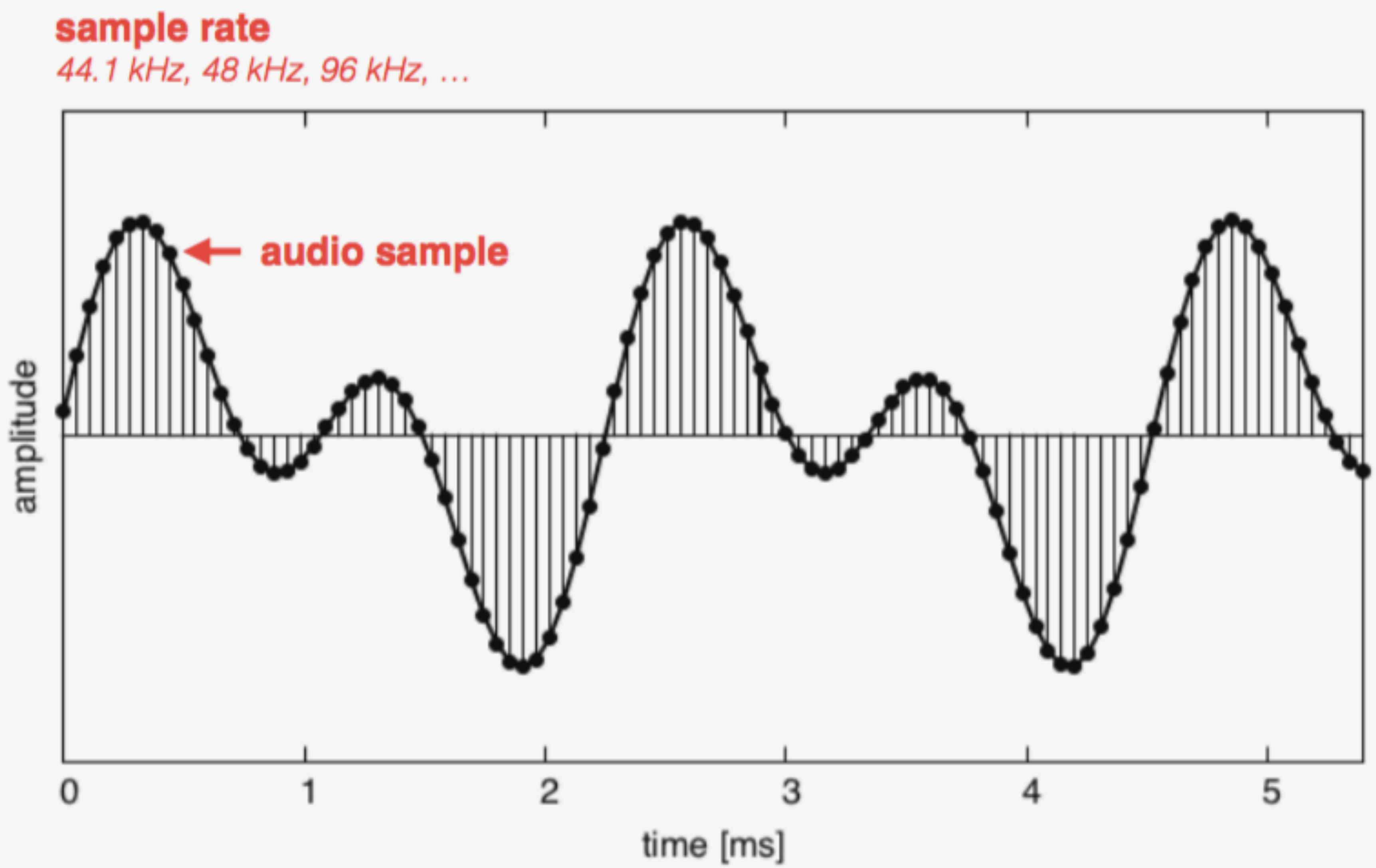


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry**. Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Amplitude Range

- Amplitude values are approximated based on the sample size / bit depth
- Audio buffers usually scale those values to floating point range [-1.0, 1.0]

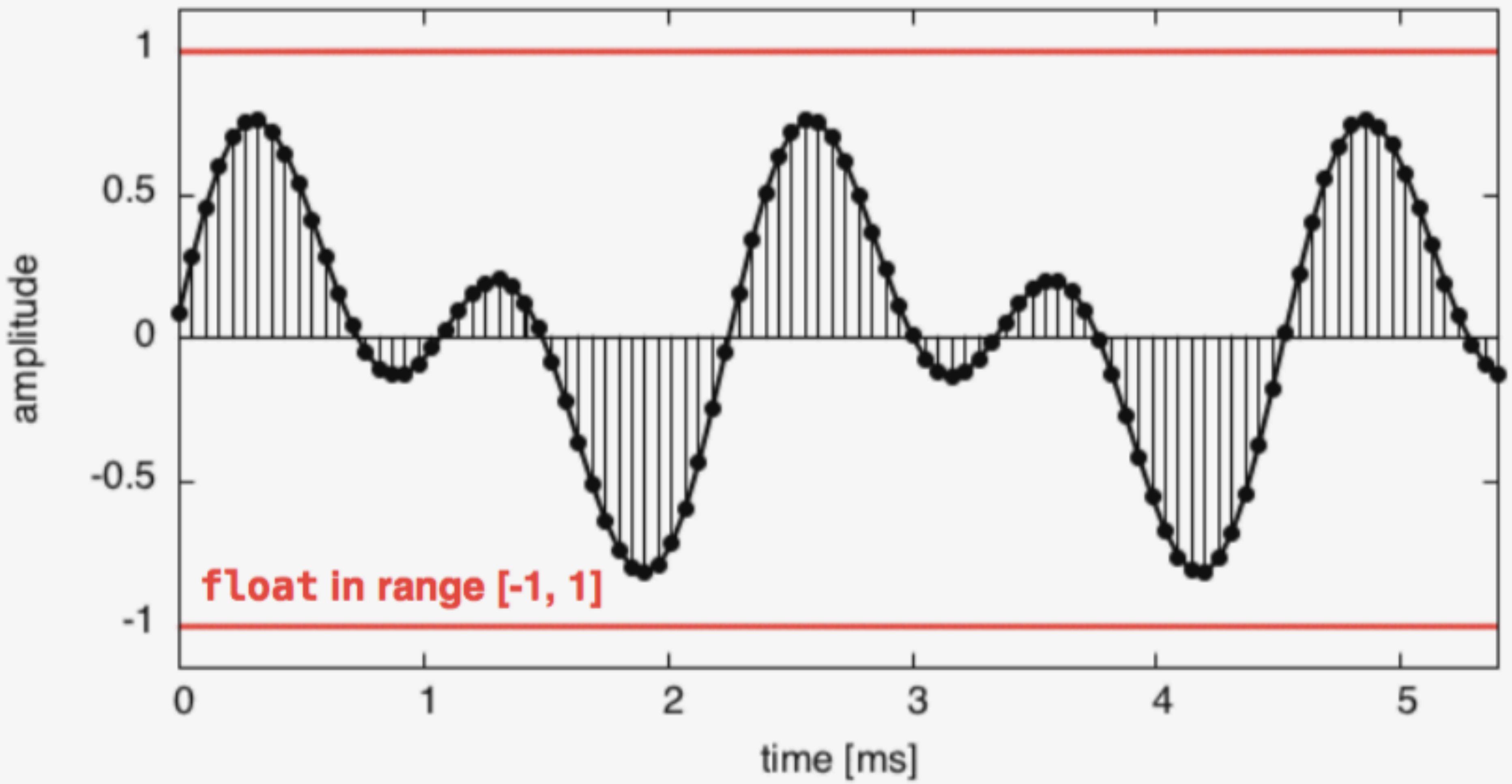


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry**. Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Channels & Frames

- Audio buffers organize amplitude values per channel (mono, stereo, …)
- An audio frame represents a (set of) sample per channel

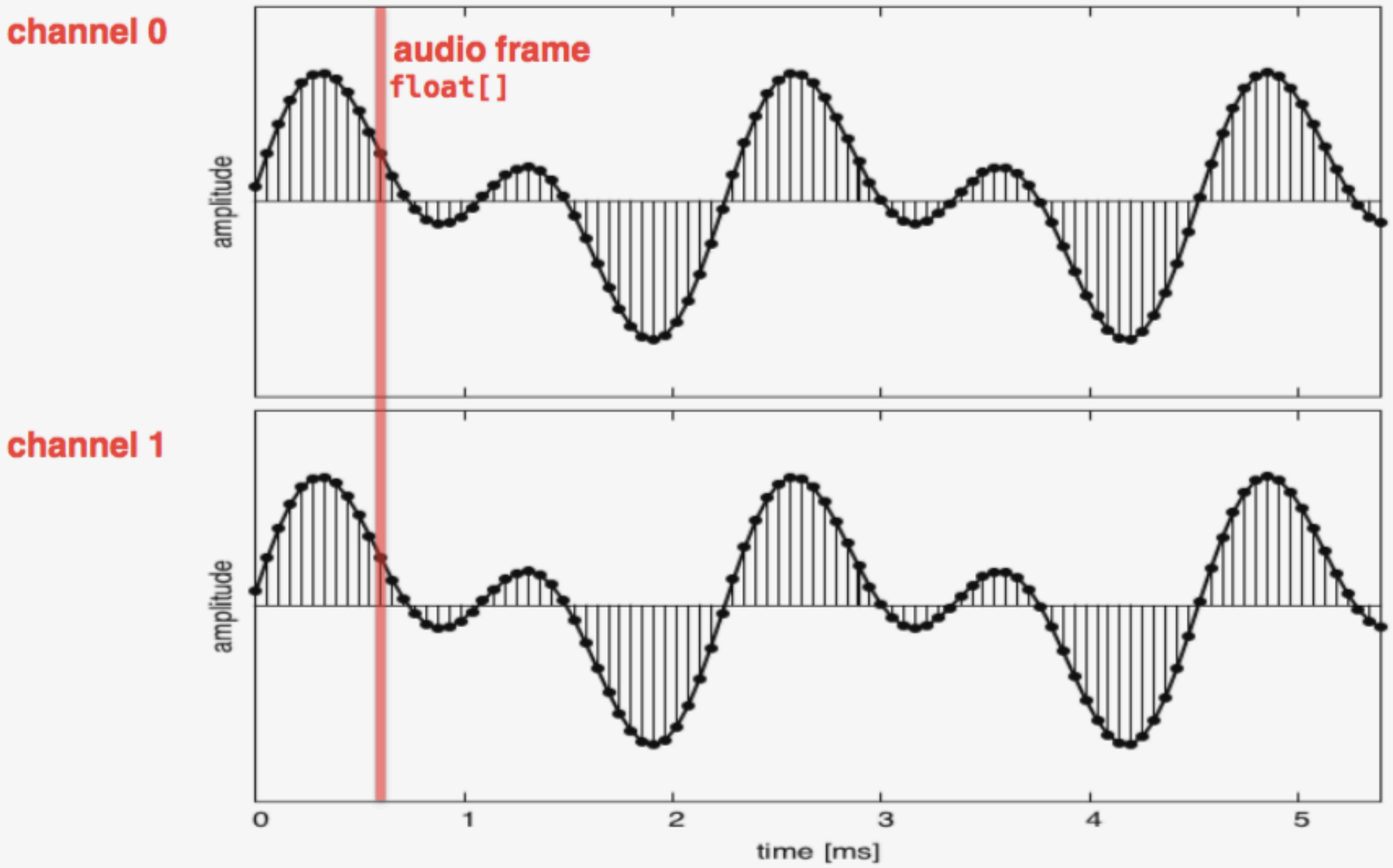


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry.** Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Audio Buffers

- Audio buffers store the sampled values per channels for all channels
- Different organization forms are used, i.e., interleaved and non-interleaved

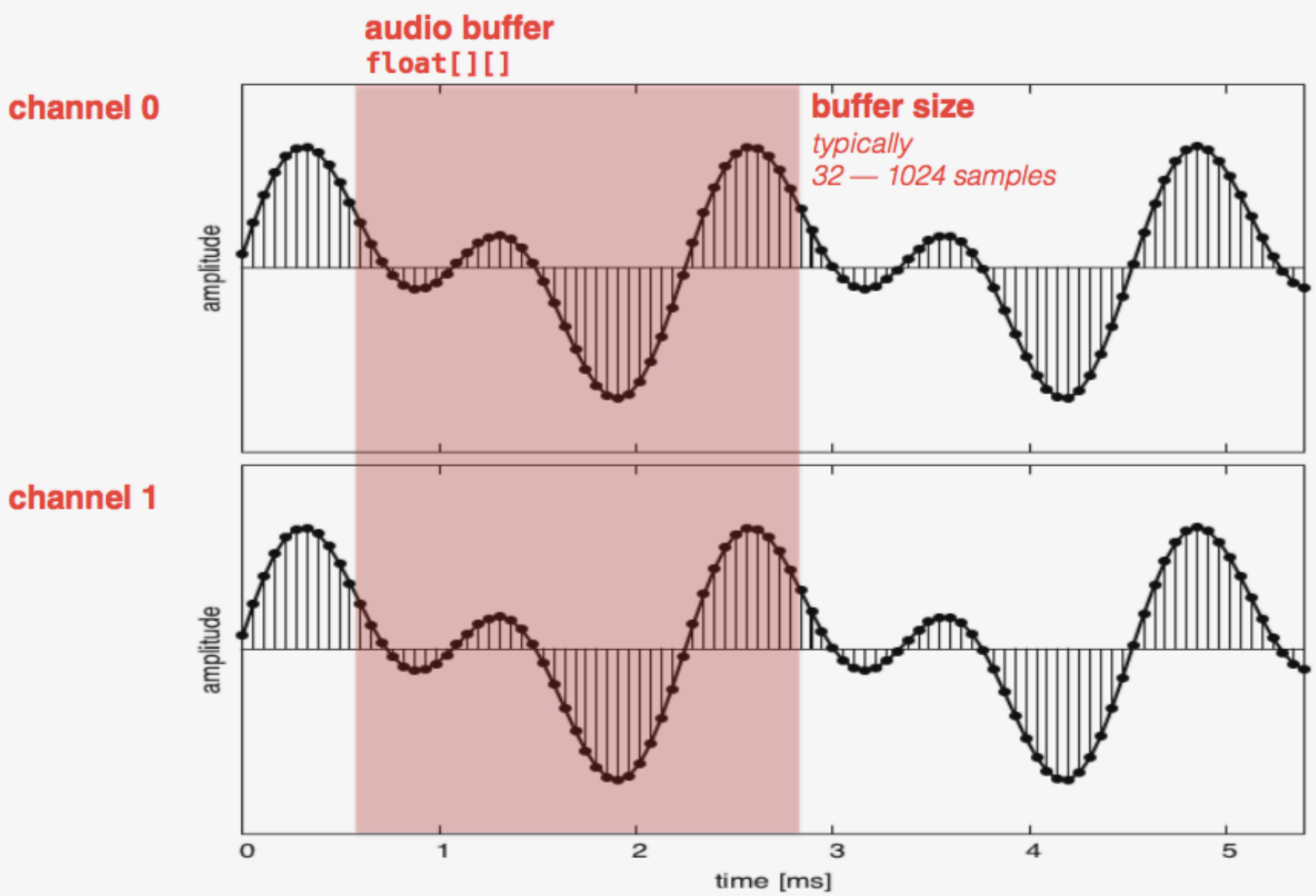


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry**. Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Audio Buffer Sizes

- Audio buffer sizes are usually kept small to ensure continuous processing of audio data streams in real-time — depending on hardware & driver capabilities

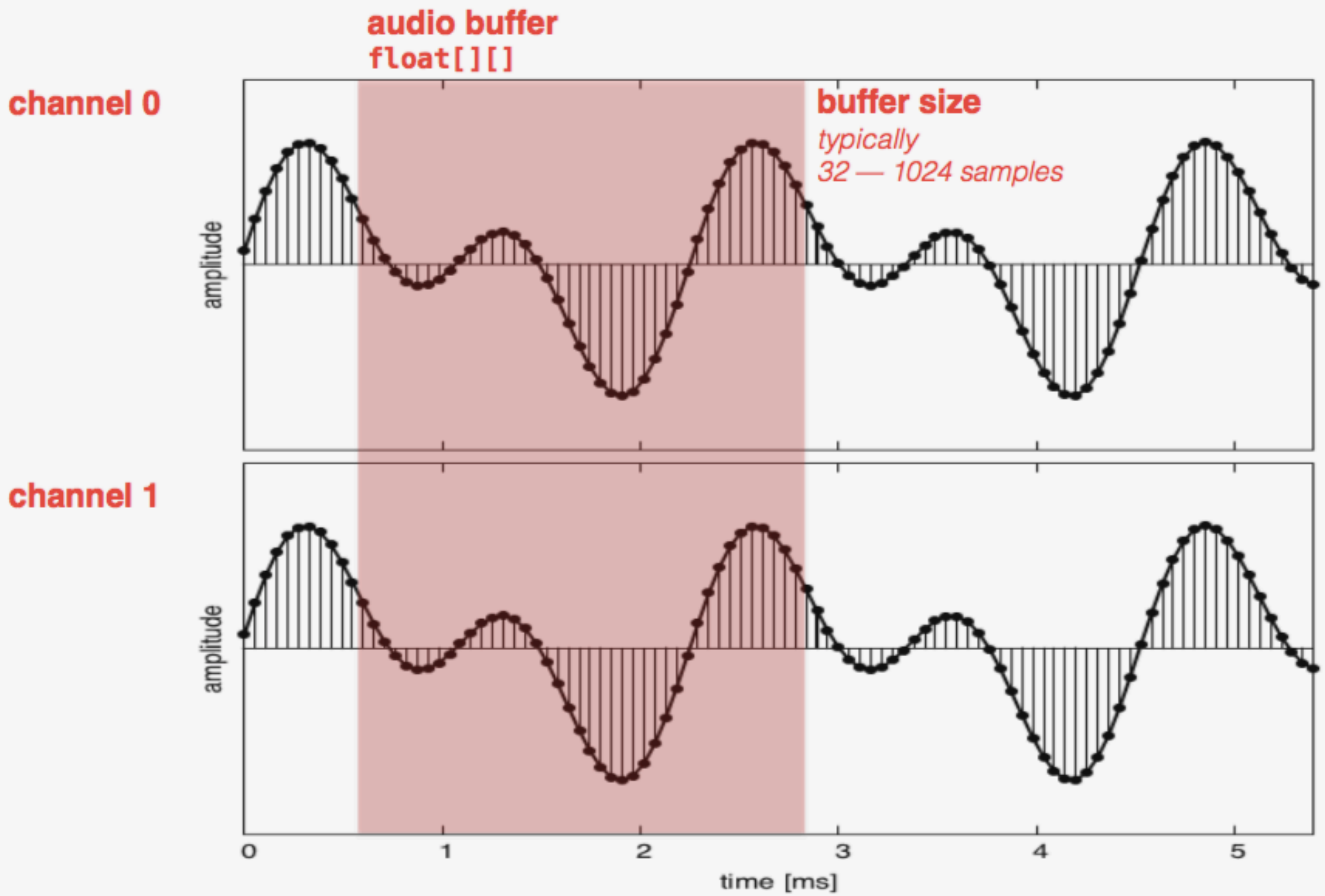


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry.** Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Real-Time Processing

- Real-time processing requires a continuous stream of data to avoid latency issues
- Latency basically describes the time delay between audio input to & output of a system

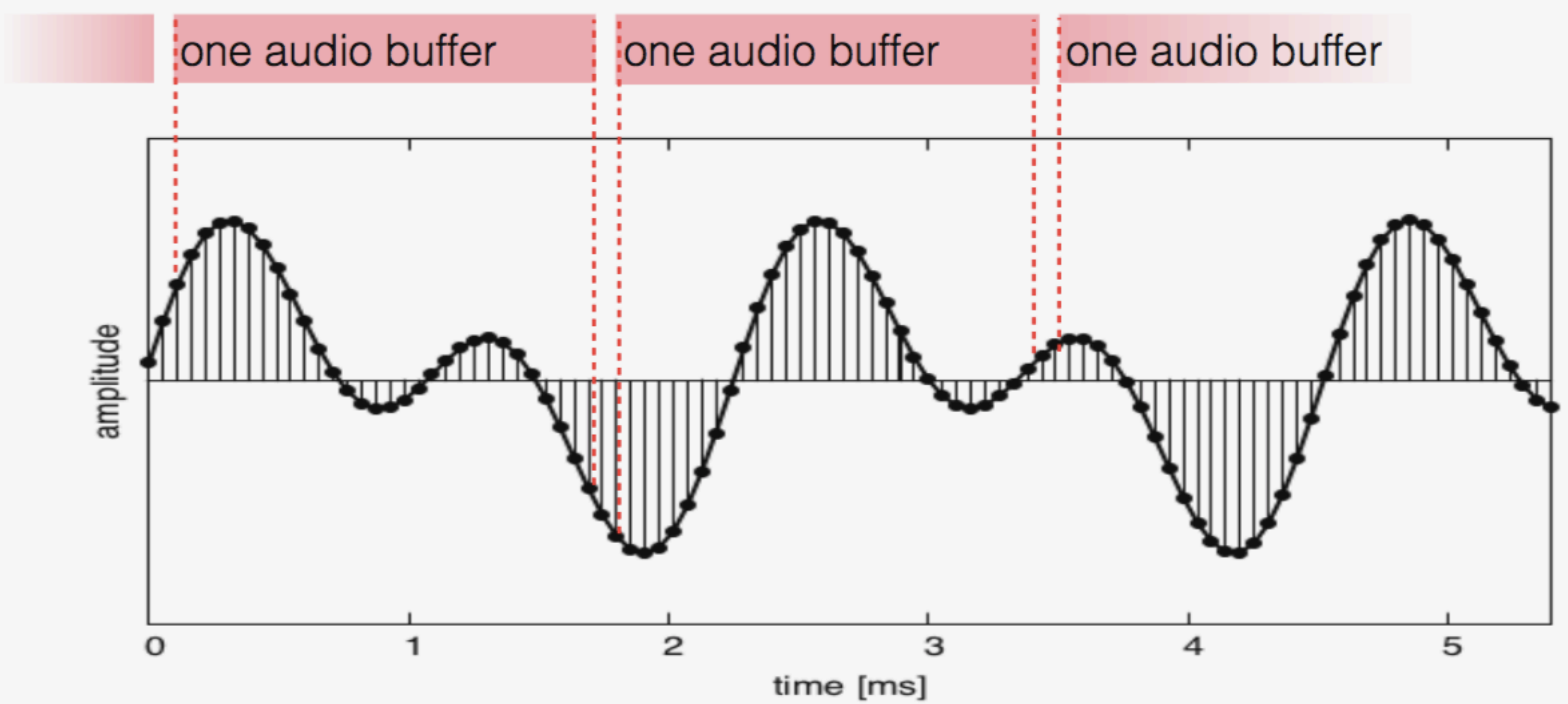


Image credit: Doumler, T. (2015): **Cpp in the Audio Industry.** Presentation at CppCon 2015. https://github.com/CppCon/CppCon2015/

Number representations in the digital domain

1. Binary
2. Integer
3. Fixed-point
4. Floating-point

Representations of vectors in IR

1. Cartesian coordinates
2. Polar coordinates
3. Transformation from Cartesian to Polar