Changing the Volume

The apparent volume of a sound is controlled by the overall amplitude of the sound itself, though the volume control on your playback device sets the maximum possible volume.

To change the overall amplitude of the sound, multiply each sample by the amount to increase or decrease it.

For example, to double the overall amplitude, multiply each sample by 2. To decrease it by 50%, multiply each sample by 0.5.

Normalizing

Normalizing is making a sound as loud as possible. We can change the intensity of the sound by increasing the amplitude of each sample by a specific amount. The amount is based on the currently largest amplitude of all the samples.

- Scan the samples array looking for the maximum amplitude.
  (assumes max is same for both positive and negative values)
- Multiply all samples by the normalization factor, the inverse of the maximum amplitude

Clipping

Clipping is a form of distortion or noise created when the absolute values of samples exceed the maximum value allowed by the system.

We can force clipping by setting all samples to their maximum value. One curious thing to notice is how well the human ear recreates the signal in the presence of so much noise.

To force clipping, set positive amplitudes to 1 and negative amplitudes to -1
Sampling Keyboards

A *sampling keyboard* makes essentially any sound an instrument by recording one tone of that sound into the keyboard’s memory, then using an algorithm to transpose that one tone to any note on the keyboard.

In practice many notes are recorded and transposed to the closest pitch.

Pitch is our perception of the *frequency* of a sound. Say we want to double the frequency. We can write a sketch that loads a sound and plays its waveform, but twice as fast as normal.

We can use a similar technique as when we scaled images.

Transposing Pitches

When scaling an image, we skipped pixels while copying. To do the same to sound samples, copy every other sample into a new sample array.

To get a new sample array, we need a new `AudioChannel` object with enough space. In Ess, to get a new channel big enough to hold every other sample of the original, do this:

```java
AudioChannel copy = new AudioChannel();
int duration = original.duration;
copy.initChannel(original.size/2, Ess.BEGINNING);
```

Doubling the frequency of a pitch makes it one *octave* higher.

Transposing Pitches

When scaling an image up, we doubled pixels while copying. To transpose a pitch an octave lower, we copy each sample twice. Recall that this requires that we add 0.5 to the index variable in the copy loop.

To hold the new sound, how big does the new channel need to be? In other words, what is the *duration* of this new sound?

Copying Short Cut

Instead of creating an empty `AudioChannel` we can simply open the sound to be modified a second time:

```java
AudioChannel copy = new AudioChannel("original.wav");
```

But if we only take half the samples, we’ll only have half the sound modified.

What will happen if we play it?

If this isn’t the effect we want, we can just *clear out* all the samples following the copied sound by setting them to zero.