An Investigation of Digital Mixing and Panning Algorithms

JESSICA KENT

SUPERVISED BY RICHARD FOSS WITH CONSULTATION FROM CORINNE COOPER

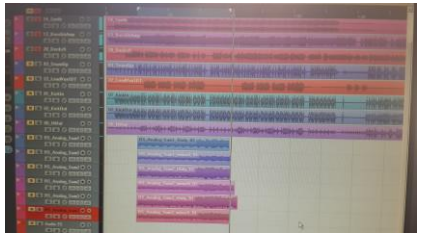
Previous Presentation

- Song is made up of multiple tracks that have been summed
- Difference between analogue and digital summing is widely debated
- Does a visual or audible difference exist?
  - Listening and visual testing
- Can digital mixing algorithm be created to emulate analogue summing?

What I have done

- Literature Review
  - Investigated source code of three DAWs
- Started programming interface to easily test and switch between audio samples
- Recorded samples in music department studio

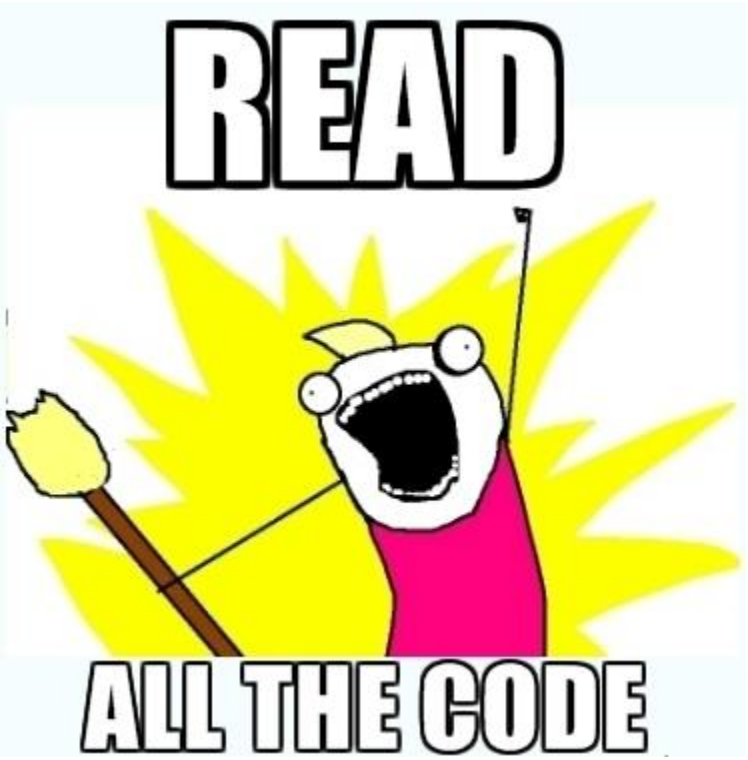
VS



Literature Review

- Investigated different aspects that could affect the digital summing process
  - Sampling Rates and Fletcher-Munson curves
  - How analogue equipment sums the tracks
  - Panning Laws
  - Summing Algorithms
- Previous testing method
  - Summing of five different DAWs tested (Leonard and Buttner-Schnirer, 2012)
    - Differences heard when panning included
- Source code

All the Source Code



Comparing the code

- Testing from 2012 investigated three specific aspects of each DAW
- Differences in representation and implementation of
  - Gain
  - Panning
  - Summing
- Ardour, Audacity and Rosegarden dealt with all three in similar ways

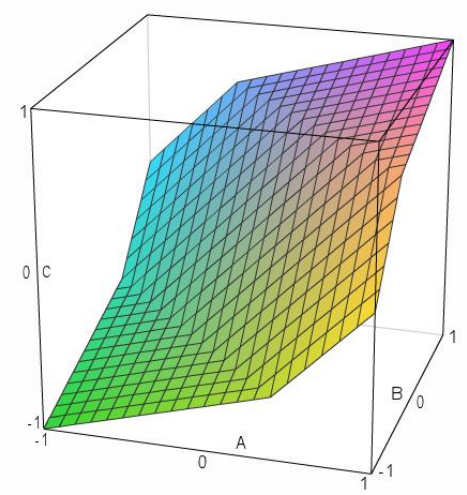
Comparing the Summing

```
void MixBuffers(int numChannels, int *channelFlags, float *gains,
                samplePtr src, samplePtr *dests,
                int len, bool interleaved)
{
        ...
      float gain = gains[c];
      float *dest = (float *)destPtr;
      float *temp = (float *)src;
      for (int j = 0; j < len; j++) {
        *dest += temp[j] * gain;    // the actual mixing process
        dest += skip;
      }
   }
```

```
void default_mix_buffers_with_gain (ARDOUR::Sample * dst,
                const ARDOUR::Sample * src,
                pframes_t nframes, float gain)
{
        for (pframes_t i = 0; i < nframes; i++) {
                dst[i] += src[i] * gain;
        }
}
void default_mix_buffers_no_gain (ARDOUR::Sample * dst,
                const ARDOUR::Sample * src, pframes_t nframes)
{
        for (pframes_t i=0; i < nframes; i++) {
                dst[i] += src[i];
        }
}
```

```
for (size_t i = 0; i < n; ++i) {
        sample_t v = cached[0][scanFrame + i]
+ cached[1][scanFrame + i];
        destination[0][i + offset] += v;
}
```

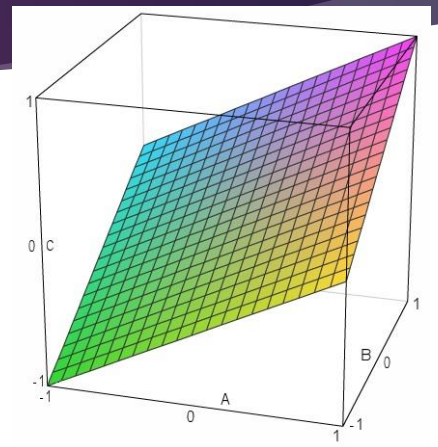
Different Summing Algorithms

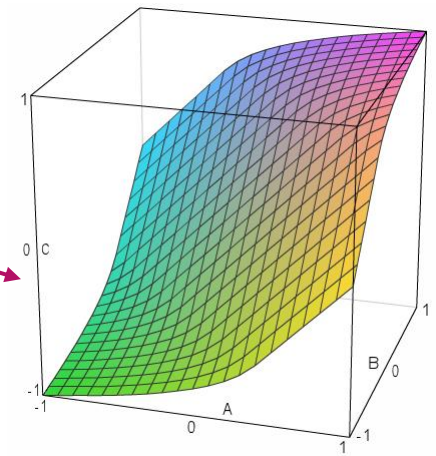


$$sum = \frac{a+b}{2}$$

$$sum = \begin{cases} x & -t \le x \le t \\ \frac{x}{|x|} \cdot (t + \frac{1-t}{2-t} \cdot (|x| - t) & |x| > t \end{cases}$$

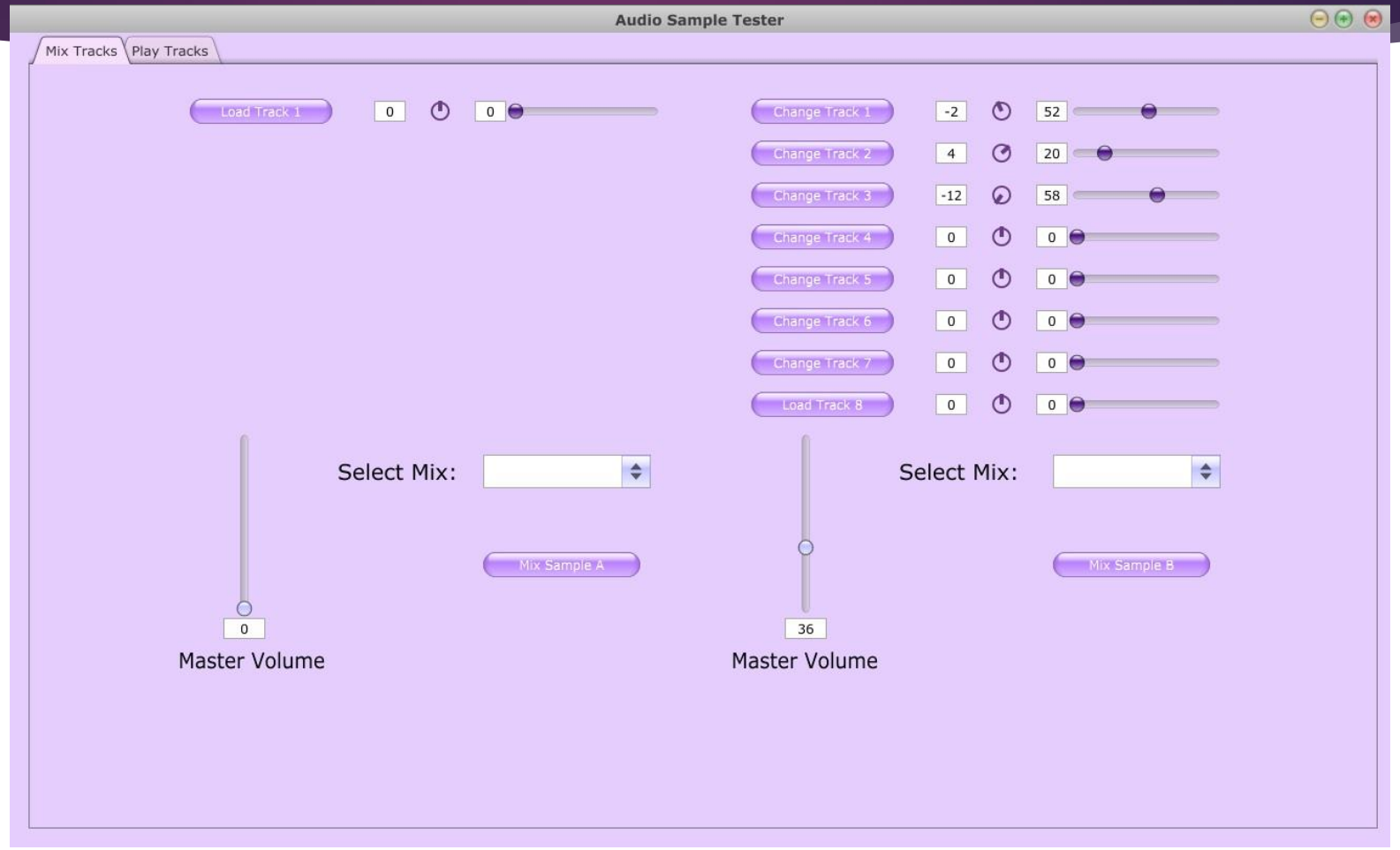
$$sum = \begin{cases} x & -t \le x \le t \\ \frac{x}{|x|} \cdot (t + (1-t) \cdot \frac{ln(1+f_\alpha(t) \cdot \frac{|x|-t}{2-t})}{ln(1+f_\alpha(t))} \cdot & |x| > t \end{cases}$$





Equations and graphs from http://voegler.eu/pub/audio/digital-audio-mixing-and-normalization.html

Audio Sample Tester



Recording

- Analogue Desk (Mackie) and Digital Workstation (Cubase)
- Summed sine waves
  - Different bit depths and sampling rates
  - How high frequency and complex waves are affected by recording

Waves	48kHz & 24-bit	48kHz & 32-bit	96kHz & 24-bit	96khz & 32-bit
30 kHz alone				
300Hz + 30kHz				
1kHz + 30kHz				
300Hz + 1kHz + 30kHz				
1kHz sin + 20kHz square				

- Summed three contemporary songs of different genres
  - Had to turn master volume down lower on digital workstation

The Rest of the Year

- Finish Audio Tester program
  - Play Tracks section of GUI
  - Code various mixing algorithms
  - If time: code different panning algorithms
- Conduct visual tests
- Conduct listening tests

QUESTIONS???