```
//We are being manipulated
//Tatiana Catanzaro
//Stanford - 220b - 2022 (Winter Quarter)

float ff_garbage_gain;

//Garbage
fun void play_garbage()
{
    SndBuf garbage => NRev reverb => dac;
    me.dir() + "audio/garbage1b.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    filename => garbage.read;
    0.1 => reverb.mix => reverb.mix;
    spork ~ fade_in10(15.0);
    while (true)
    {
        1.6 * ff_garbage_gain => garbage.gain;
        100::ms => now;
    }
}

fun void fade_in10(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    0.25/duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
    {
        swell_up.last() => ff_garbage_gain;
        10::ms => now;
    }
}

//Garbage4
fun void play_garbage4()
{
    SndBuf garbage4 => dac;
    me.dir() + "audio/garbage4.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    4.5 => garbage4.gain;
    filename => garbage4.read;
    1::day => now;
}

//Garbage Mujica
fun void play_garbagemujica()
{
```

```
    SndBuf garbage2 => dac;
    me.dir() + "audio/mujica-under-garbage2.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    2.3 => garbage2.gain;
    filename => garbage2.read;
    1::day => now;
}

//Horn
fun void play_horn()
{
    SndBuf horn => NRev reverb => dac;
    me.dir() + "audio/horn.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    1.5 => horn.gain;
    filename => horn.read;
    0.1 => reverb.mix => reverb.mix;
    1::day => now;
}

0.0 => float pan_noiserec;
0.0 => float ff_noiserec_gain;

fun void noise_record()
{
    SndBuf noise => Pan2 p => dac;
    me.dir() + "audio/play-noise.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    filename => noise.read;
    spork ~pan_noiserecord(30.0);
    spork ~ fade_innoise(15.0);
    while ( true )
    {
        pan_noiserec => p.pan;
        3.5 * ff_noiserec_gain => noise.gain;
        10::ms => now;
    }
}

fun void pan_noiserecord(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    5::second => now;
    1./duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
    {
        swell_up.last() => pan_noiserec;
```

```
        1::ms => now;
    }
}


fun void fade_innoise(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    0.25/duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
    {
        swell_up.last() => ff_noiserec_gain;
        10::ms => now;
    }
}

/*
0.0 => float ff_gain2;

//Noise
fun void play_noise()
{
    SndBuf garbage => Envelope e => BiQuad f => dac;
    me.dir() + "audio/garbage4.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    filename => garbage.read;
    dac => Delay delay => dac;
    spork ~fade_in2(0.0, 0.03, 8.0);
    // our radius
    .99999 => float R;
    // our delay order
    500 => float L;
    // set delay
    L::samp => delay.delay;
    // set dissipation factor
    Math.pow( R, L ) => delay.gain;
    // set the filter's pole radius
    .99 => f.prad;
    // set equal gain zeros
    1 => f.eqzs;
    // initialize float variable
    0.0 => float v;
    // set filter gain
    //.05 => f.gain;
    // infinite time-loop
    while( true )
    {
```

```
        Math.random2f (10,500)::ms => dur t => e.duration;
        // key on - start attack
        e.keyOn();
        // advance time by 800 ms
        900::ms => now;
        // key off - start release
        e.keyOff();
        // sweep the filter resonant frequency
        Std.fabs(Math.sin(v)) * 4000.0 => f.pfreq;
        // increment v
        v + .1 => v;
        ff_gain2 => e.gain;
        1::ms => now;
    }
}

fun void fade_in2(float start2, float end2, float total_time2)
{
    now + total_time2::second => time end_time2;
    while ( now < end_time2)
    {
        ((end2 - start2) / total_time2) / 1000.0 => float per_ms2;
        per_ms2 +=> ff_gain2;
        1::ms => now;
    }
}
*/


//money
fun void money()
{
    SndBuf buf2 => dac;
    me.dir() + "/audio/word-money.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    0.3 => buf2.rate;
    1::day => now;
}

fun void money2()
{
    SndBuf buf2 => dac;
    me.dir() + "/audio/word-money.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    0.35 => buf2.rate;
    1::day => now;
}
```

```
fun void money3()
{
    SndBuf buf2 => dac;
    me.dir() + "/audio/word-money.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    0.4 => buf2.rate;
    1::day => now;
}

fun void money4()
{
    SndBuf buf2 => dac;
    me.dir() + "/audio/word-money.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    1 => buf2.rate;
    1::day => now;
}

//Making money
fun void play_buf2()
{
    SndBuf buf2 => dac;
    me.dir() + "/audio/making-money.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    0.85 => buf2.gain;
    1::day => now;
}

//Become a Dasher
fun void play_buf3()
{
    SndBuf buf2 => dac;
    me.dir() + "/audio/Become-a-Dasher-cut.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    0.75 => buf2.gain;
    1::day => now;
}

//Rolling coins
fun void play_buf4()
{
    SndBuf buf2 => NRev reverb => dac;
    me.dir() + "/audio/rolling-coins.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
```

```
    1.5 => buf2.gain;
    filename1 => buf2.read;
    0.0075 => reverb.mix => reverb.mix;
    1::day => now;
}

0.0 => float ff_4_2_gain;

fun void play_buf4_2()
{
    SndBuf buf4_2 => NRev reverb => dac;
    me.dir() + "/audio/rolling-coins.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf4_2.read;
    0.085 => reverb.mix => reverb.mix;
    spork ~ fade_in4_2(8.0);
    while (true)
    {
        1 * ff_4_2_gain => buf4_2.gain;
        100::ms => now;
    }
    //let it ring
    while( true )
    {
        ff_4_2_gain => buf4_2.gain;
        1::ms => now;
    }
}

fun void fade_in4_2(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    0.25/duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
    {
        swell_up.last() => ff_4_2_gain;
        10::ms => now;
    }
}

//Carlin advertisement

0.0 => float ff_gain_carlin;

fun void play_buf5()
{
    SndBuf buf2 => NRev reverb => dac;
```

```
    me.dir() + "/audio/carlin-advertisement-cut.wav" => string filename1;
    if( me.args() ) me.arg(0) => filename1;
    filename1 => buf2.read;
    0.085 => reverb.mix => reverb.mix;
    spork ~fade_in_carlin(3.0);
    1::day => now;
    while (true)
    {
        6 * ff_gain_carlin => reverb.gain;
        100::ms => now;
    }
}

fun void fade_in_carlin(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    0.25/duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
    {
        swell_up.last() => ff_gain_carlin;
        10::ms => now;

    }
}

0.0 => float ff_gain_buf6;

fun void play_buf6()
{
    SndBuf buf6 => Gain node => dac;
    node => Delay delay => Gain atten => node;
    me.dir() + "/audio/carlin-advertisement-cut2.wav" => string filename6;
    if( me.args() ) me.arg(0) => filename6;
    filename6 => buf6.read;
    0.3 => node.gain;
    0.2::second => delay.max => delay.delay;
    0.5 => atten.gain;
    1::day => now;
    while( true )
    {
        4 * ff_gain_buf6 => node.gain;
        1::ms => now;
    }

}

fun void fade_in_buf6 (float start, float end, float total_time)
```

```
{
    now + total_time::second => time end_time;
    while ( now < end_time)
    {
        ((end - start) / total_time) / 1000.0 => float per_ms_buf6;
        per_ms_buf6 +=> ff_gain_buf6;
        1::ms => now;
    }
}

0.0 => float ff_gain;

//making money2
fun void feedforward()
{
    me.dir() + "/audio/making-money.wav" => string filename;
    SndBuf buf => Gain node => dac;
    //feedback
    node => Delay delay2 => Gain atten => node;
    filename => buf.read;
    //set delay capacity and delay time
    0.2::second => delay2.max => delay2.delay;
    // attenuate the feedback volume
    .85 => atten.gain;
    0.85 => buf.rate;
    spork ~fade_in(0.0, 0.6, 2.0);
    //let it ring
    while( true )
    {
        ff_gain => node.gain;
        1::ms => now;
    }

}

fun void fade_in (float start, float end, float total_time)
{
    now + total_time::second => time end_time;
    while ( now < end_time)
    {
        ((end - start) / total_time) / 1000.0 => float per_ms;
        per_ms +=> ff_gain;
        1::ms => now;
    }
}

0.0 => float pan_feed2;

fun void feedforward2_record()
```

```
{
    SndBuf feed => Pan2 p => dac;
    me.dir() + "audio/feedforward2.wav" => string filename;
    if( me.args() ) me.arg(0) => filename;
    2.3 => feed.gain;
    filename => feed.read;
    spork ~pan_feedforward2(30.0);
    while ( true )
    {
        0.6 * pan_feed2 => p.pan;
        10::ms => now;
    }
}

fun void pan_feedforward2(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    5::second => now;
    0.25/duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
    {
        swell_up.last() => pan_feed2;
        1::ms => now;
    }
}


0.0 => float ff2_gain;

//making money2
fun void feedforward2()
{
    me.dir() + "/audio/feedforward.wav" => string filename;
    SndBuf buf => Gain node => dac;
    //feedback
    node => Delay delay2 => Gain atten => node;
    filename => buf.read;
    //set delay capacity and delay time
    0.17::second => delay2.max => delay2.delay;
    // attenuate the feedback volume
    .8 => atten.gain;
    0.95 => buf.rate;
    spork ~fade_in(0.0, 0.24, 5.0);
    //let it ring
    while( true )
    {
        0.02 * ff2_gain => node.gain;
```

```
        1::ms => now;
    }

}

fun void fade_in (float start, float end, float total_time)
{
    now + total_time::second => time end_time;
    while ( now < end_time)
    {
        ((end - start) / total_time) / 1000.0 => float per_ms;
        per_ms +=> ff2_gain;
        1::ms => now;
    }
}

// rolling coins
0.0 => float pan_lot;

fun void lottery()
{
    me.dir() + "/audio/rolling-coins.wav" => string filename2;
    SndBuf buf => Gain node => Pan2 p => dac;
    spork ~pan_lottery(10.0);
    //feedback
    node => Delay delay => Gain atten => node;
    filename2 => buf.read;
    //set delay capacity and delay time
    0.3::second => delay.max => delay.delay;
    // attenuate the feedback volume
    .9=> atten.gain;
    //let it ring
    while ( true )
    {
        pan_lot => p.pan;
        1::ms => now;
    }
}

fun void pan_lottery(float time_second)
{
    SinOsc swell_up => blackhole;
    time_second => float duration_sec;
    5::second => now;
    0.25/duration_sec => swell_up.freq;
    1000::ms => now;
    1 - 0.25/duration_sec => swell_up.freq;
    now + time_second::second => time finish;
    while (now < finish)
```

```
    {
        swell_up.last() => pan_lot;
        1::ms => now;
    }
}

fun void accumulation()
{
    me.dir() + "/audio/rolling-coins.wav" => string filename2;
    SndBuf buf => Gain node => dac;
    //feedback
    node => Delay delay => Gain atten => node;
    filename2 => buf.read;
    //set delay capacity and delay time
    1::second => delay.max => delay.delay;
    // attenuate the feedback volume
    .7 => atten.gain;
    //let it ring
    while( true )
        1::day => now;
}

0.0 => float ff_gain_acc2;

fun void accumulation2()
{
    me.dir() + "/audio/rolling-coins.wav" => string filename2;
    SndBuf buf => Gain node => ResonZ f => dac;
    //feedback
    node => Delay delay => Gain atten => node;
    filename2 => buf.read;
    //set delay capacity and delay time
    .05::second => delay.max => delay.delay;
    // attenuate the feedback volume
    .981 => atten.gain;
    // set filter Q (higher == narrower, sharper resonance)
    2 => f.Q;
    // sweep the cutoff
    100 + Math.fabs(Math.sin(now/second)) * 4000 => f.freq;
    spork ~fade_in_acc2(0.0, 1.001, 5.0);
    //let it ring
    while( true )
    {
        ff_gain_acc2 => node.gain;
        1::ms => now;
    }
}

fun void fade_in_acc2 (float start, float end, float total_time)
```

```
{
    now + total_time::second => time end_time;
    while ( now < end_time)
    {
        ((end - start) / total_time) / 1000.0 => float per_ms;
        per_ms +=> ff_gain_acc2;
        1::ms => now;
    }
}

fun void accumulation3()
{
    me.dir() + "/audio/rolling-coins.wav" => string filename2;
    SndBuf buf => Gain node => ResonZ f => dac;
    //feedback
    node => Delay delay => Gain atten => node;
    filename2 => buf.read;
    1.015 => node.gain;
    //set delay capacity and delay time
    .1::second => delay.max => delay.delay;
    // attenuate the feedback volume
    .9 => atten.gain;
    // set filter Q (higher == narrower, sharper resonance)
    2 => f.Q;
    // sweep the cutoff
    100 + Math.fabs(Math.sin(now/second)) * 3000 => f.freq;
    // advance time
    //let it ring
    while( true )
        1::ms => now;
}

fun void phase_changer()
{
    spork ~play_buf2();
    0.3::second => now;
    spork ~play_buf3();
    0.8::second => now;
    spork ~money();
    2.3::second => now;
    spork ~money2();
    spork ~play_buf2();
    1.5::second => now;
    spork ~play_buf4();
    spork ~feedforward();
    0.5::second => now;
    spork ~feedforward2();
    1.2::second => now;
    spork ~feedforward2_record();
```

```
    spork ~money();
    3::second => now;
    spork ~accumulation();
    1::second => now;
    spork ~money3();
    spork ~play_buf4_2();
    spork ~accumulation();
    0.5::second => now;
    spork ~money();
    2::second => now;
    spork ~accumulation2();
    3.5::second => now;
    spork ~lottery();
    0.5::second => now;
    spork ~money4();
    0.3::second => now;
    spork ~feedforward2();
    3::second => now;
    spork ~play_garbage();
    2.5::second => now;
    spork ~lottery();
    1.5::second => now;
    spork ~play_buf5();
    6::second => now;
    spork ~accumulation3();
    4::second => now;
    spork ~play_horn();
    28::second => now;
    //spork ~play_noise();
    spork ~noise_record();
    spork ~play_garbage4();
    20::second => now;
    spork ~play_garbagemujica();


}

phase_changer();
// advance time
3::minute => now;
```