```
// adapted code from GE Wang and also "Programming for musi
// authors Ajay Kapur, Perry Cook, Spencer Salazar and Ge W

// z axis deadzone
0 => float DEADZONE;

// map values to variables
0 => float xaxis;
0 => float yaxis;
0 => float zaxis;
0 => float xrotation;
0 => float yrotation;
0 => float zrotation;

// which joystick
0 => int device;

// get from command line
if( me.args() ) me.arg(0) => Std.atoi => device;

// HID objects
Hid trak;
HidMsg msg;

// open joystick 0, exit on fail
if( !trak.openJoystick( device ) ) me.exit();

// print
<<< "joystick '" + trak.name() + "' ready", "" >>>;

SinOsc xleft => dac;
1 => int onGainXleft;
0 => int offGainXleft;

SinOsc yleft => dac;
1 => int onGainYleft;
```

```
0 => int offGainYleft;

SinOsc zleft => dac;
1 => int onGainZleft;
0 => int offGainZleft;

SinOsc xright => dac;
1 => int onGainXright;
0 => int offGainXright;

SinOsc yright => dac;
1 => int onGainYright;
0 => int offGainYright;

SinOsc zright => dac;
1 => int onGainZright;
0 => int offGainZright;

SinOsc button => dac;
1 => int onGainButton;
0 => int offGainButton;


// data structure for gametrak
class GameTrak
{
    // timestamps
    time lastTime;
    time currTime;

    // previous axis data
    float lastAxis[6];
    // current axis data
    float axis[6];
}
```

```
// gametrack
GameTrak gt;


// spork control
spork ~ gametrak();
// print
spork ~ print();

// main loop
while( true )
{
    100::ms => now;
}

// print
fun void print()
{
    // time loop
    while( true )
    {
        // left controller values gt.axis[0] is x-axis, gt.
        // right controller values gt.axis[3] is x-rotation
        // the minimum and maximum values need to be determ
        <<< "axes:", gt.axis[0],gt.axis[1]*100,gt.axis[2], 
        // advance time
        //1000::ms => now;
        100::ms => now;
    }
}

// gametrack handling
fun void gametrak()
{

    while( true )
```

```
{
    // wait on HidIn as event
    trak => now;

    // messages received
    while( trak.recv( msg ) )
    {
        // joystick axis motion
        if( msg.isAxisMotion() )
        {
            // check which
            if( msg.which >= 0 && msg.which < 6 )
            {
                // check if fresh
                if( now > gt.currTime )
                {
                    // time stamp
                    gt.currTime => gt.lastTime;
                    // set
                    now => gt.currTime;

                }
                // save last
                gt.axis[msg.which] => gt.lastAxis[msg.wh
                // the z axes map to [0,1], others map t
                if( msg.which != 2 && msg.which != 5 )
                { msg.axisPosition => gt.axis[msg.which]
                else
                {
                    1 - ((msg.axisPosition + 1) / 2) - [
                    if( gt.axis[msg.which] < 0 ) 0 => gt
                }
            }

            // check value for x-axis and map to a tone
            if(gt.axis[0] < 0.0){
```

```
    261.6 => xleft.freq;
}else if(gt.axis[0] > 80.0){
    17.3 => xleft.freq;
}else{
    493.9 => xleft.freq;
}

// check value for y-axis and map to a tone
if(gt.axis[1] < 0){
    19.4 => yleft.freq;
}else if(gt.axis[1] > 80.0){
    21.8 => yleft.freq;
}else{
    20.6 => yleft.freq;
}

// check value for z-axis and map to a tone
if(gt.axis[2] < 0){
    65.4 => zleft.freq;
}else if(gt.axis[2] > 0.5){
    110.0 => zleft.freq;
}else{
    349.2 => zleft.freq;
}

// check value for x-rotation and map to a
if(gt.axis[3] < 0.0){
    43.6 => xright.freq;
}else if(gt.axis[3] > 80.0){
    698.5 => xright.freq;
}else{
    174.6 => xright.freq;
}

// check value for y-rotation and map to a
if(gt.axis[4] < 0){
```

```
        19.4 => yright.freq;
    }else if(gt.axis[4] > 80.0){
        21.8 => yright.freq;
    }else{
        20.6 => yright.freq;
    }

    // check value for z-rotation and map to a f
    if(gt.axis[5] < 0){
        65.4 => zright.freq;
    }else if(gt.axis[5] > 0.5){
        110.0 => zright.freq;
    }else{
        349.2 => zright.freq;
    }

    onGainXleft => xleft.gain;
    0.1 :: second => now;
//    onGainYleft => yleft.gain;
//    0.1 :: second => now;
    onGainZleft => zleft.gain;
    0.1 :: second => now;
    offGainXleft => xleft.gain;
    0.1 :: second => now;
    offGainYleft => yleft.gain;
    0.1 :: second => now;
    offGainZleft => zleft.gain;
    0.1 :: second => now;

    onGainXright => xright.gain;
    0.1 :: second => now;
    onGainYright => yright.gain;
    0.1 :: second => now;
    onGainZright => zright.gain;
    0.1 :: second => now;
    offGainXright => xright.gain;
```

```
                    0.1 :: second => now;
                    offGainYright => yright.gain;
                    0.1 :: second => now;
                    offGainZright => zright.gain;
                    0.1 :: second => now;
                    // joystick button down
                    if( msg.isButtonDown() )
                    {
                        <<< "button", msg.which, "down" >>>;
                        onGainButton => button.gain;
                        0.3 :: second => now;
                        offGainButton => button.gain;
                        0.3 :: second => now;
                    }
                }

            }
        }
    }
```

axis[0] = left x -1.0 to 1.0
axis[1] = left y -1.0 to 1.0
axis[2] = left string pull
axis[3] = right x -1.0 to 1.0
axis[4] right y -1.0 to 1.0
axis[5] = right string pull
0 to 1
isButtonDown()
isButtonUp()
true false

```
// name: gametra.ck
// desc: gametrak boilerplate example
//
// author: Ge Wang (ge@ccrma.stanford.edu)
// date: summer 2014

// z axis deadzone
0 => float DEADZONE;

// which joystick
0 => int device;
// get from command line
if( me.args() ) me.arg(0) => Std.atoi => device;

// HID objects
Hid trak;
HidMsg msg;

// open joystick 0, exit on fail
if( !trak.openJoystick( device ) ) me.exit();

// print
<<< "joystick '" + trak.name() + "' ready", "" >>>;

// data structure for gametrak
class GameTrak
{
    // timestamps
    time lastTime;
    time currTime;

    // previous axis data
    float lastAxis[6];
    // current axis data
    float axis[6];
}
```

261.6 - middle C
293.7 - D
329.6 - E

```
// gametrack
GameTrak gt;


// spork control
spork ~ gametrak();
// print
spork ~ print();

// main loop
while( true )
{
    100::ms => now;
}

// print
fun void print()
{
    // time loop
    while( true )
    {
        // values
        <<< "axes:", gt.axis[0],gt.axis[1],gt.axis[2], gt.a
        // advance time
        100::ms => now;
    }
}

// gametrack handling
fun void gametrak()
{
    while( true )
    {
        // wait on HidIn as event
        trak => now;
```

```
// messages received
while( trak.recv( msg ) )
{
    // joystick axis motion
    if( msg.isAxisMotion() )
    {
        // check which
        if( msg.which >= 0 && msg.which < 6 )
        {
            // check if fresh
            if( now > gt.currTime )
            {
                // time stamp
                gt.currTime => gt.lastTime;
                // set
                now => gt.currTime;
            }
            // save last
            gt.axis[msg.which] => gt.lastAxis[msg.wh
            // the z axes map to [0,1], others map t
            if( msg.which != 2 && msg.which != 5 )
            { msg.axisPosition => gt.axis[msg.which]
            else
            {
                1 - ((msg.axisPosition + 1) / 2) - l
                if( gt.axis[msg.which] < 0 ) 0 => gt
            }
        }
    }

    // joystick button down
    else if( msg.isButtonDown() )
    {
        <<< "button", msg.which, "down" >>>;
    }
```

```
            // joystick button up
            else if( msg.isButtonUp() )
            {
                <<< "button", msg.which, "up" >>>;
            }
        }
    }
}
```