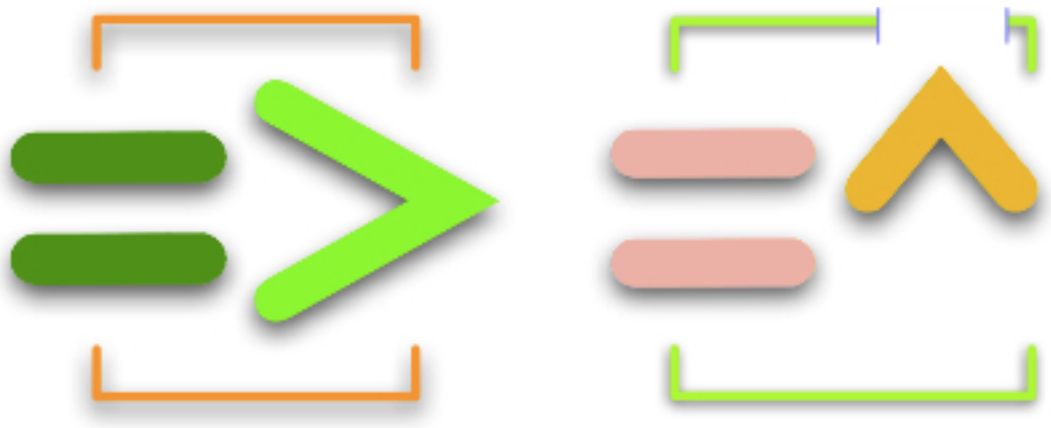
소프트웨어학부

CSE2020 음악프로그래밍



2
Libraries:
ChucK’s built-in tools

ChucK Standard Library
`Std`

See

`https://chuck.cs.princeton.edu/doc/program/stdlib.html`

or
Appendix B
in text

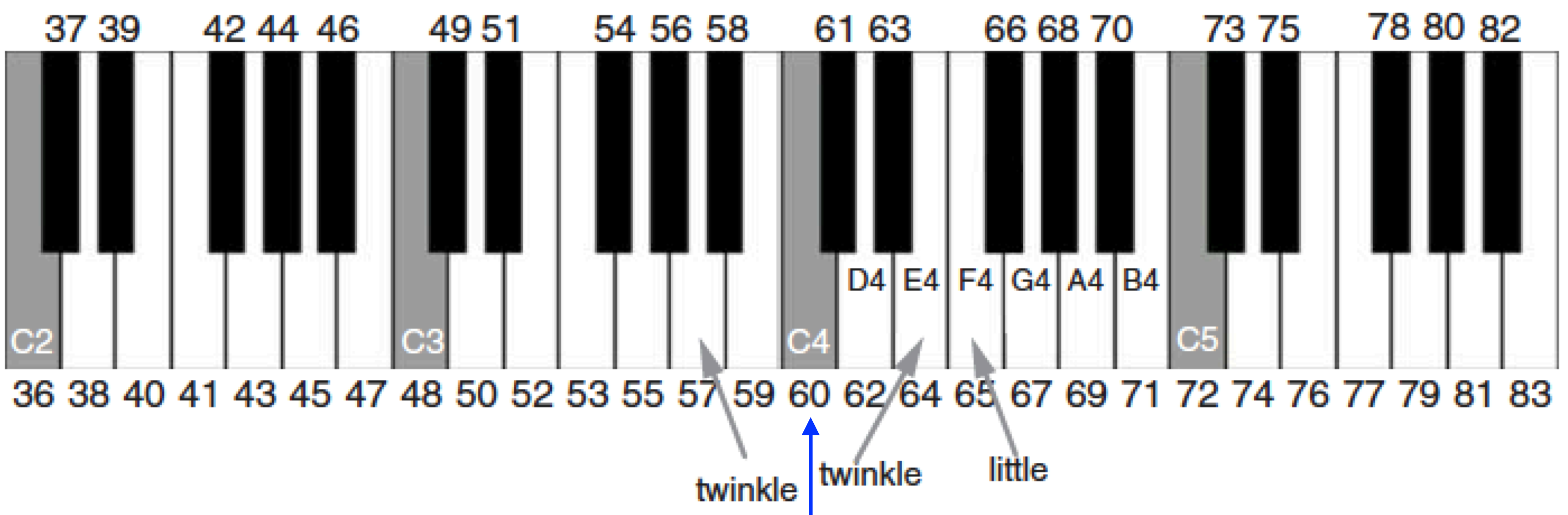
ChucK Standard Library

MIDI note numbers

Musical
Instrument
Digital
Interface

0~127

piano keyboard
musical keyboard

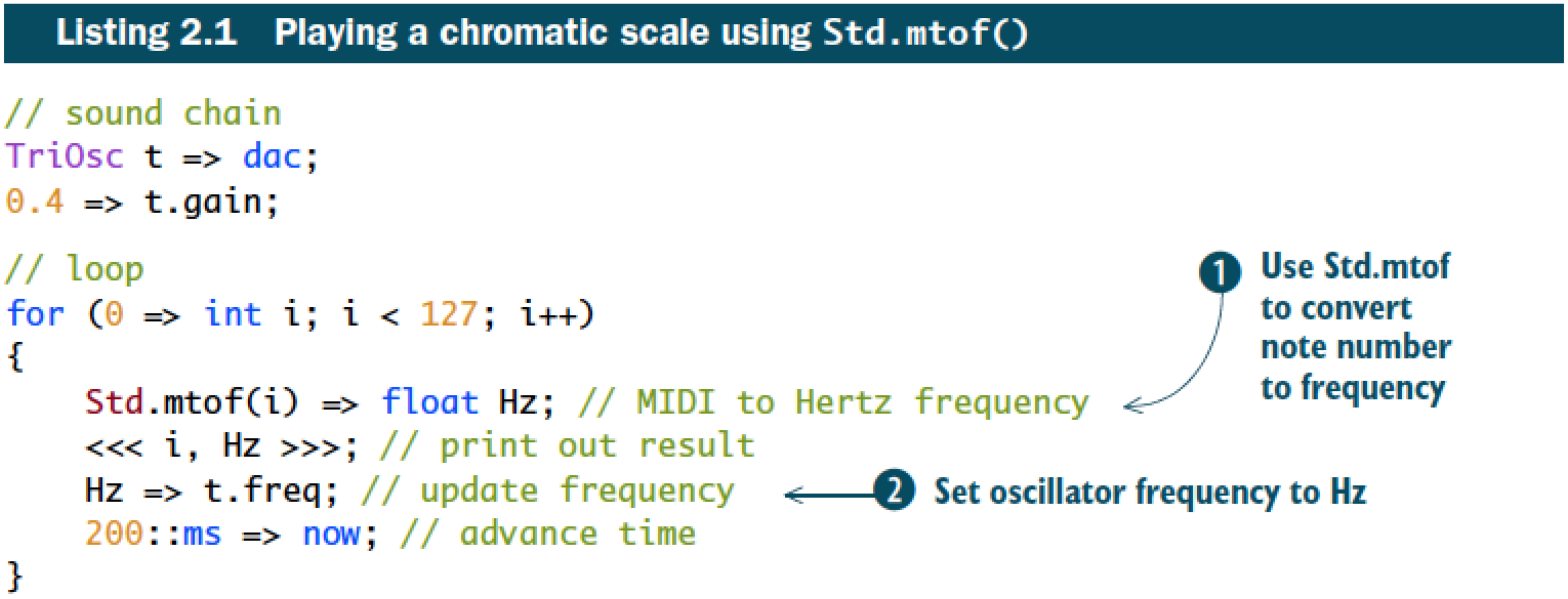


middle C

Function name and arguments	What it does
`float mtof(float value);`	Converts a MIDI note number to frequency (Hz). Note the input value is of type `float` (supports fractional note number).
`float ftom(float value);`	Converts frequency (Hz) to MIDI note number space.

```
<<< Std.mtof(64) >>>;
<<< 64 => Std.mtof >>>;
<<< Std.mtof(60), Std.mtof(62), Std.mtof(64), Std.mtof(65), Std.mtof(67) >>>;
```

ChucK Standard Library



Listing 2.1 Playing a chromatic scale using `Std.mtof()`

```
// sound chain
TriOsc t => dac;
0.4 => t.gain;

// loop
for (0 => int i; i < 127; i++)
{
    Std.mtof(i) => float Hz; // MIDI to Hertz frequency
    <<< i, Hz >>>; // print out result
    Hz => t.freq; // update frequency
    200::ms => now; // advance time
}
```

1 Use Std.mtof to convert note number to frequency

2 Set oscillator frequency to Hz

- Run this code to figure out the range of frequencies you can hear!
- Rewrite the code using `while` loop instead of `for` loop

ChucK Standard Library

ChucK as a language is strongly typed.

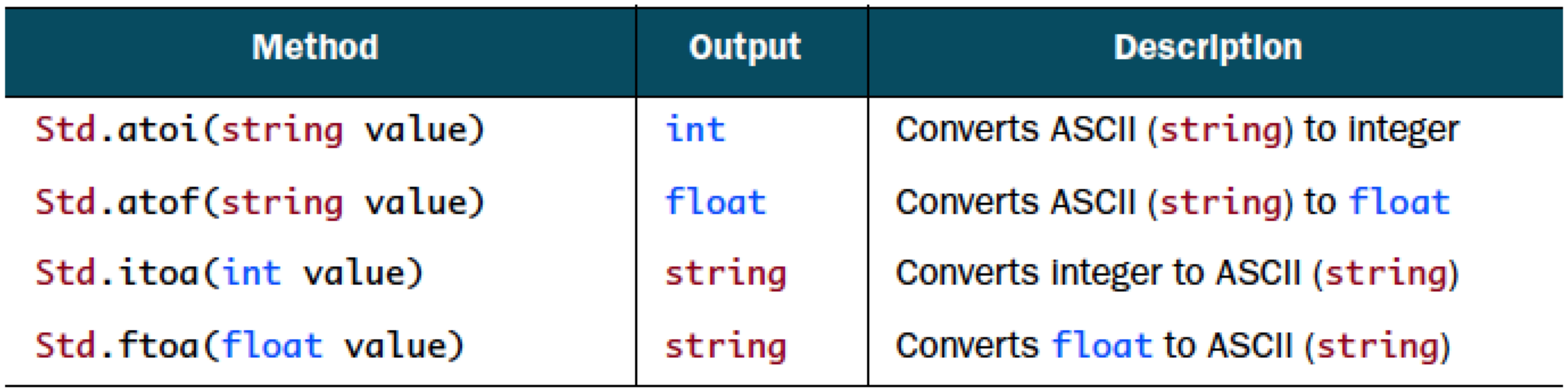
Std functions converting and dealing with basic data types

Function name and arguments	What the function does
`int abs(int value);`	Returns absolute value of integer.
`float fabs(float value);`	Returns absolute value of floating point number.
`float sgn(float value);`	Computes sign of input as -1.0 (neg), 0, or 1.0 (pos).
`int ftoi(float value);`	Converts floating-point number to integer (by truncation).

```
220.0 => int myFreq;    X
  220 => int myFreq;    O

myFreq => float myFloatFreq;    O

220.5 => Std.ftoi => int myFreq;    O

Std.ftoi(220.5) => int myFreq;    O        0.5 is thrown away!
```

`Std` functions converting between string and number

```
Std.atoi("128.7");
```



Method	Output	Description
`Std.atoi(string value)`	`int`	Converts ASCII (`string`) to integer
`Std.atof(string value)`	`float`	Converts ASCII (`string`) to `float`
`Std.itoa(int value)`	`string`	Converts integer to ASCII (`string`)
`Std.ftoa(float value)`	`string`	Converts `float` to ASCII (`string`)

ChucK Math Library
`Math`

See

https://chuck.cs.princeton.edu/doc/program/stdlib.html

or
Appendix B
in text

ChucK Math Library

Random number generation

Listing 2.2 Random integer generation using the Math library

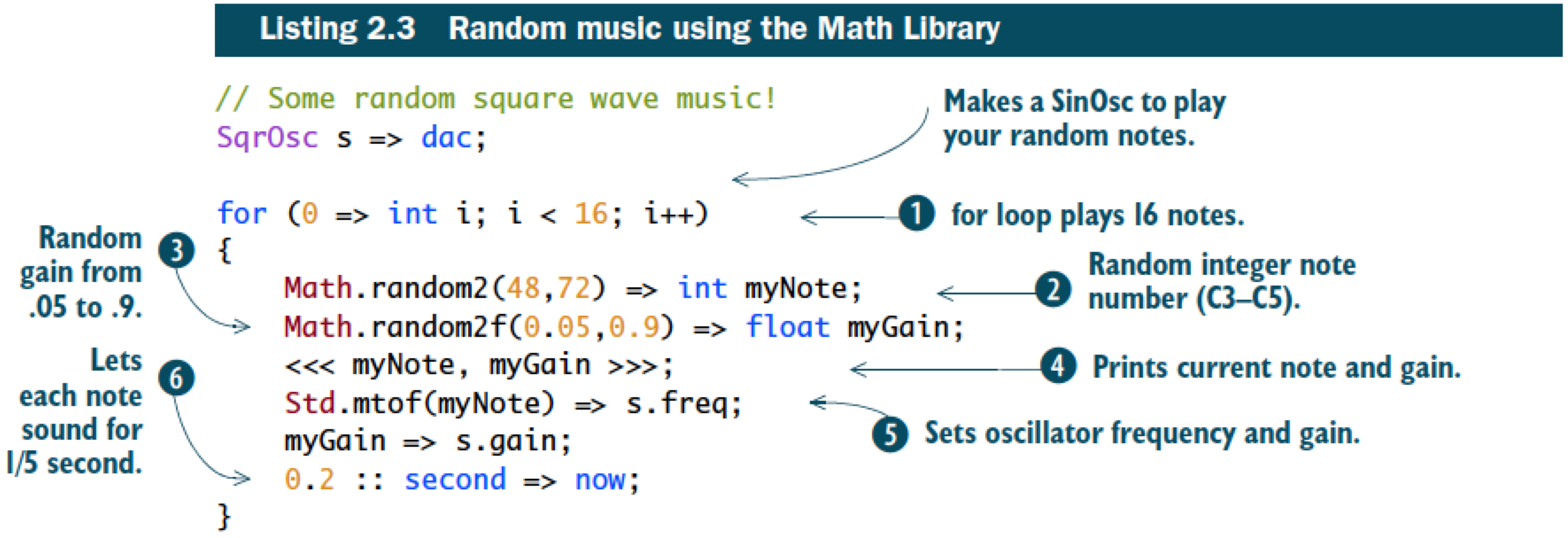
```
// random integer number generation
// simulates the roll of a die
while (true)
{
    <<< "Dice Roll =", Math.random2(1,6) >>>;
    second / 2 => now;
}
```

Table 2.2 Chuck Math library functions to create random numbers

Method	Output	Description
`Math.random()`	`int`	Generates random integer between 0 and Math.RANDOM_MAX
`Math.random2(int min, int max)`	`int`	Generates random integer in the range [min, max]
`Math.randomf()`	`float`	Generates random floating point number in the range [0, 1]
`Math.random2f(float min, float max)`	`float`	Generates random floating point number in the range [min, max]

ChucK Math Library

Random number generation



Listing 2.3 Random music using the Math Library

```
// Some random square wave music!
SqrOsc s => dac;

for (0 => int i; i < 16; i++)
{
    Math.random2(48,72) => int myNote;
    Math.random2f(0.05,0.9) => float myGain;
    <<< myNote, myGain >>>;
    Std.mtof(myNote) => s.freq;
    myGain => s.gain;
    0.2 :: second => now;
}
```

Makes a SinOsc to play your random notes.

1 for loop plays 16 notes.

2 Random integer note number (C3–C5).

3 Random gain from .05 to .9.

4 Prints current note and gain.

5 Sets oscillator frequency and gain.

6 Lets each note sound for 1/5 second.

Random number generation with seed

```
Math.srandom(134);
```

ChucK Math Library

Rounding numbers

being more fair about float-to-int conversion

```
<<< Math.round(220.501), Math.round(220.49) >>>;
```

Stereo and Panning

panorama = view of a region surrounding an observer

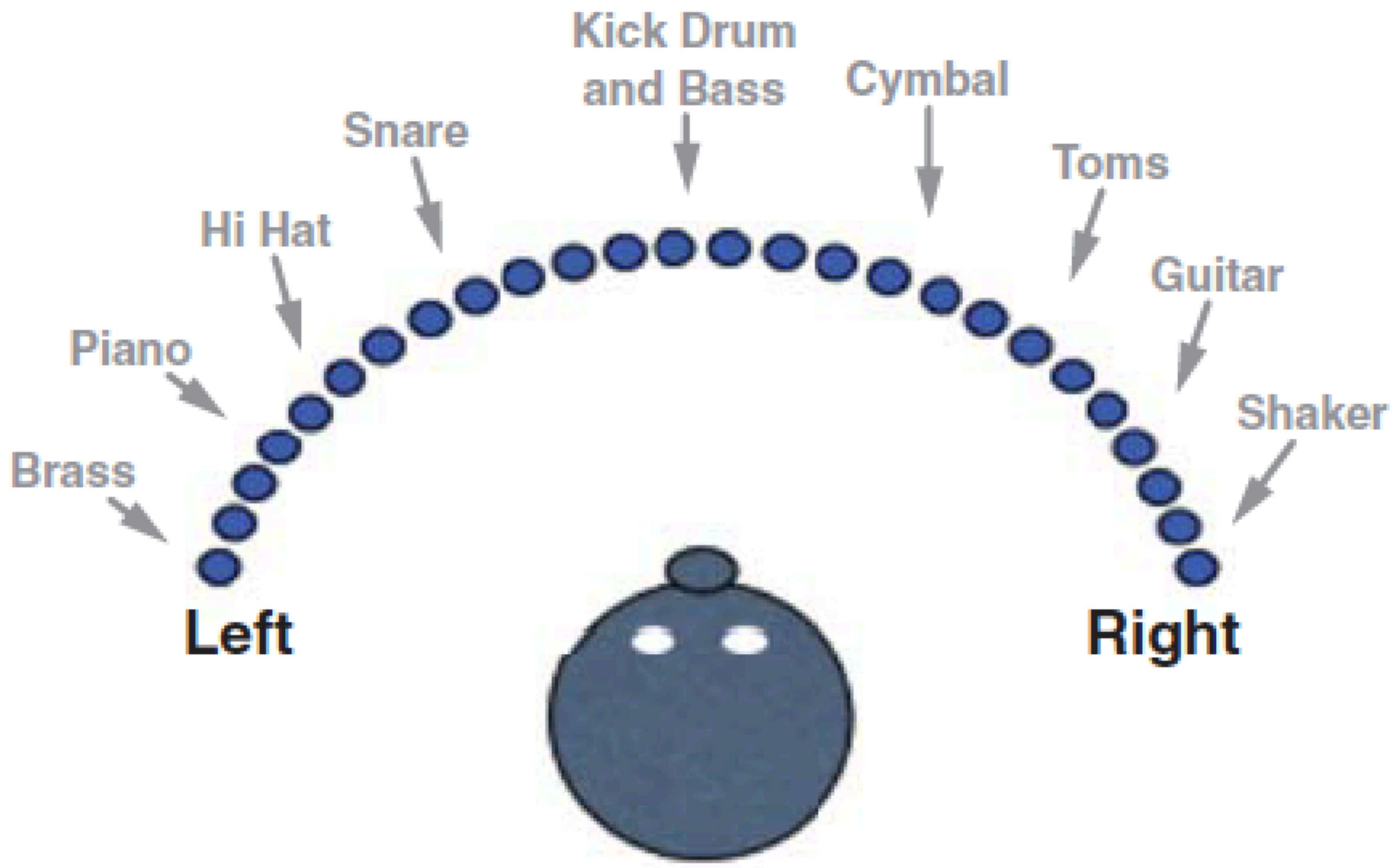
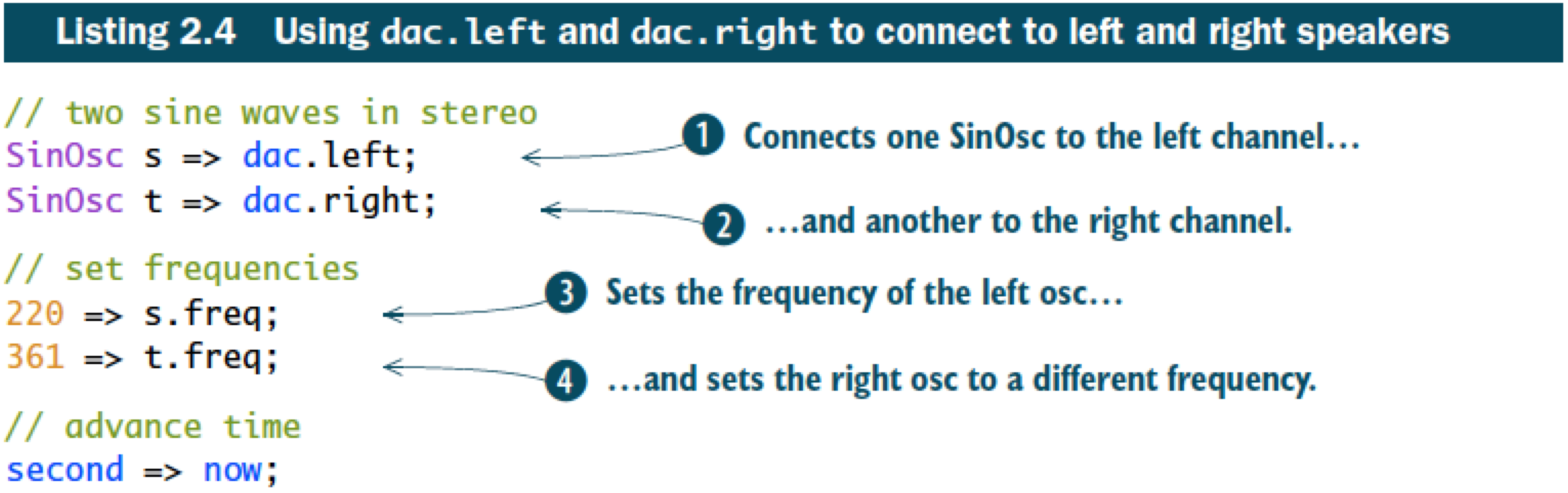


Figure 2.2 **Panning is used to position the instruments in a stereo sound field.**

Panning Technique #1



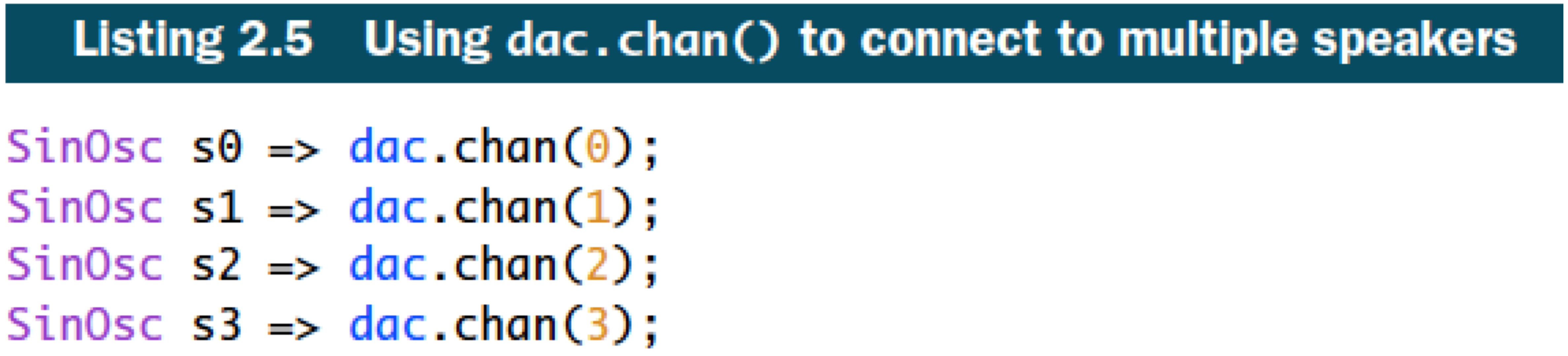
Listing 2.4 Using `dac.left` and `dac.right` to connect to left and right speakers

```
// two sine waves in stereo
SinOsc s => dac.left;
SinOsc t => dac.right;

// set frequencies
220 => s.freq;
361 => t.freq;

// advance time
second => now;
```

1. Connects one SinOsc to the left channel…
2. …and another to the right channel.
3. Sets the frequency of the left osc…
4. …and sets the right osc to a different frequency.

Stereo and Panning

Panning Technique #2

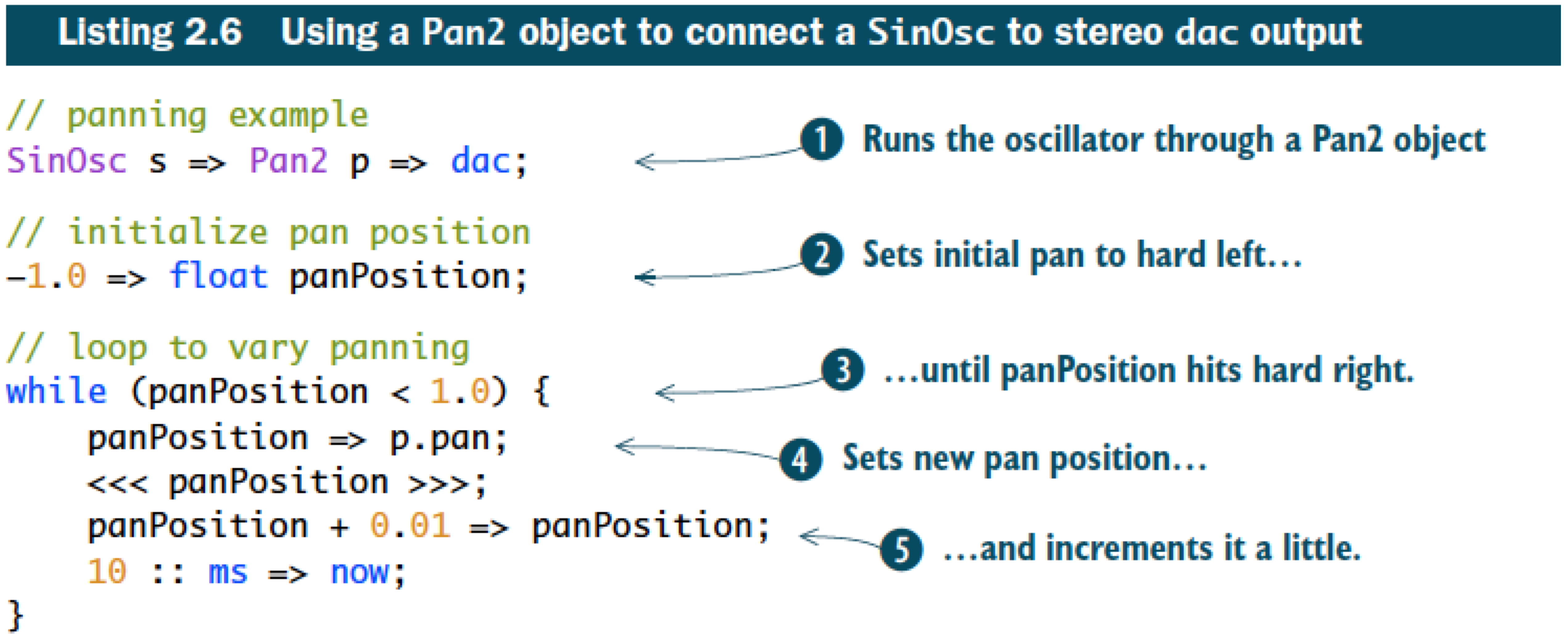


Listing 2.5 Using dac.chan() to connect to multiple speakers

```
SinOsc s0 => dac.chan(0);
SinOsc s1 => dac.chan(1);
SinOsc s2 => dac.chan(2);
SinOsc s3 => dac.chan(3);
```

Stereo and Panning

Panning Technique #3



Listing 2.6 Using a Pan2 object to connect a SinOsc to stereo *dac* output

```
// panning example
SinOsc s => Pan2 p => dac;

// initialize pan position
-1.0 => float panPosition;

// loop to vary panning
while (panPosition < 1.0) {
    panPosition => p.pan;
    <<< panPosition >>>;
    panPosition + 0.01 => panPosition;
    10 :: ms => now;
}
```

1. Runs the oscillator through a Pan2 object
2. Sets initial pan to hard left…
3. …until panPosition hits hard right.
4. Sets new pan position…
5. …and increments it a little.

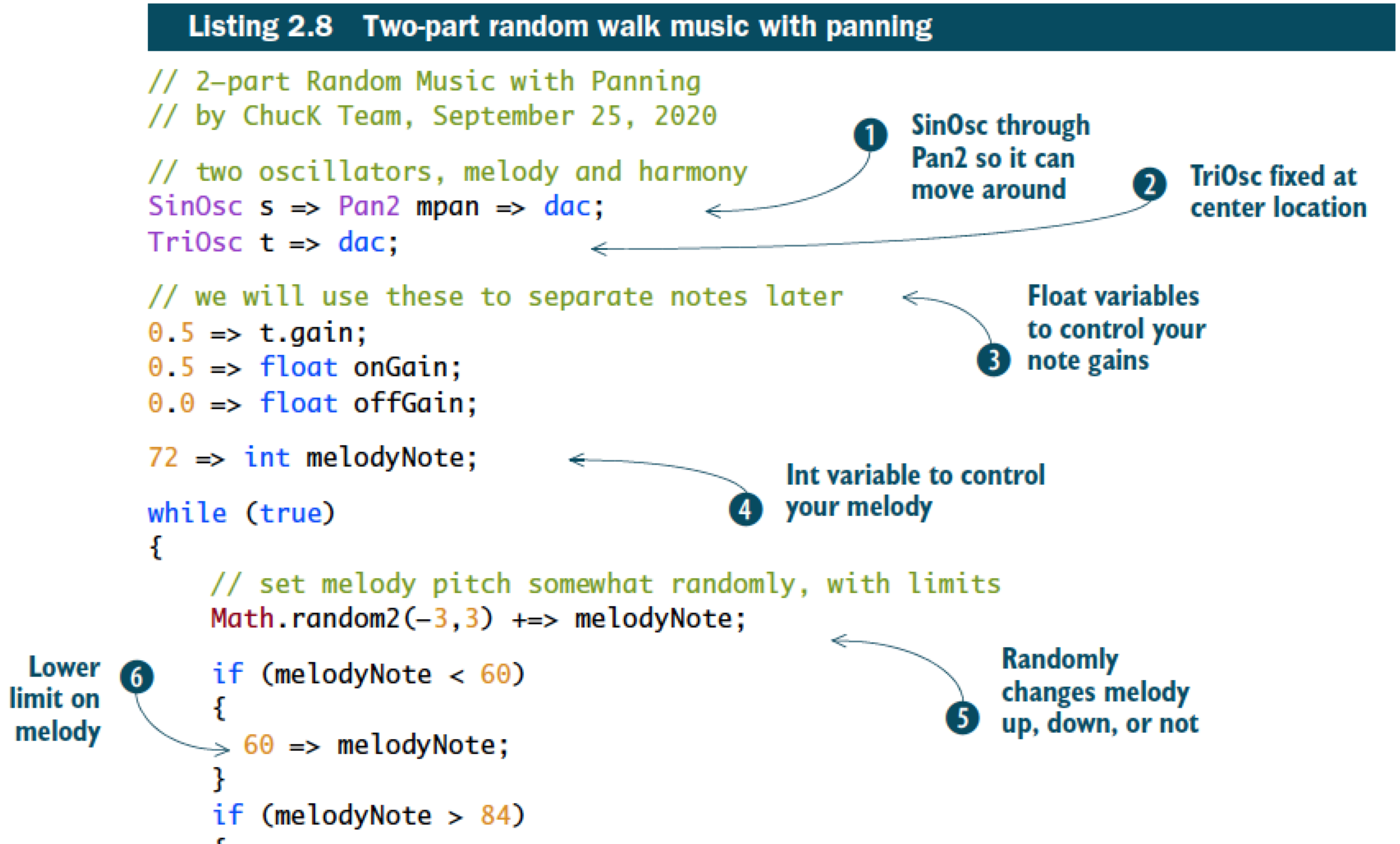
Panning Technique #3

Listing 2.7 Automatic panning using Pan2 and the Math.sin() function

```
//sound chain: white noise to pan2 to dac
Noise n => Pan2 p => dac;

//noise can sound quite loud
0.2 => n.gain;

// infinite loop
while (true)
{
    //oscillate pan between 1.0 and -1.0
    Math.sin(now/second) => p.pan;
    //do it pretty often, to make it smooth
    ms => now;
}
```

Random Music with Voices and Panning

Example



Listing 2.8 Two-part random walk music with panning

```
// 2-part Random Music with Panning
// by ChucK Team, September 25, 2020

// two oscillators, melody and harmony
SinOsc s => Pan2 mpan => dac;
TriOsc t => dac;

// we will use these to separate notes later
0.5 => t.gain;
0.5 => float onGain;
0.0 => float offGain;

72 => int melodyNote;

while (true)
{
    // set melody pitch somewhat randomly, with limits
    Math.random2(-3,3) +=> melodyNote;

    if (melodyNote < 60)
    {
        60 => melodyNote;
    }
    if (melodyNote > 84)
```

1. SinOsc through Pan2 so it can move around
2. TriOsc fixed at center location
3. Float variables to control your note gains
4. Int variable to control your melody
5. Randomly changes melody up, down, or not
6. Lower limit on melody

Int variable to control your melody (4)

```
while (true)
{
    // set melody pitch somewhat randomly, with limits
    Math.random2(-3,3) +=> melodyNote;

    if (melodyNote < 60)
    {
     60 => melodyNote;
    }
    if (melodyNote > 84)
    {
     84 => melodyNote;
    }

    Std.mtof(melodyNote) => s.freq;

    // melody has a random pan for each note
    Math.random2f(-1.0,1.0) => mpan.pan;

    // On a "dice roll," change harmony note
    if (Math.random2(1,6) == 1)
    {
        Std.mtof(melodyNote-12) => t.freq;
    }

    // Pick one of three random durations
    Math.random2(1,3)*0.2 => float myDur;

    // note on time is 80% of duration
    onGain => s.gain;
    (myDur*0.8)::second => now;

    // space between notes is 20% of array duration
    offGain => s.gain;
    (myDur*0.2)::second => now;
}
```

(5) Randomly changes melody up, down, or not

(6) Lower limit on melody

(7) Upper limit on melody

(8) Sets solo SinOsc pitch

(9) Randomly sticks TriOsc on a pitch