MUSC 208 Winter 2014
John Ellinger, Carleton College

Lab 5

Laptop Setup to Record Audio from ChucK into Audacity

Do this lab on your laptop. Follow these instructions on the course web page.

MAC
http://acad.carleton.edu/courses/musc208-00-w14/MIDIsetupMac.html

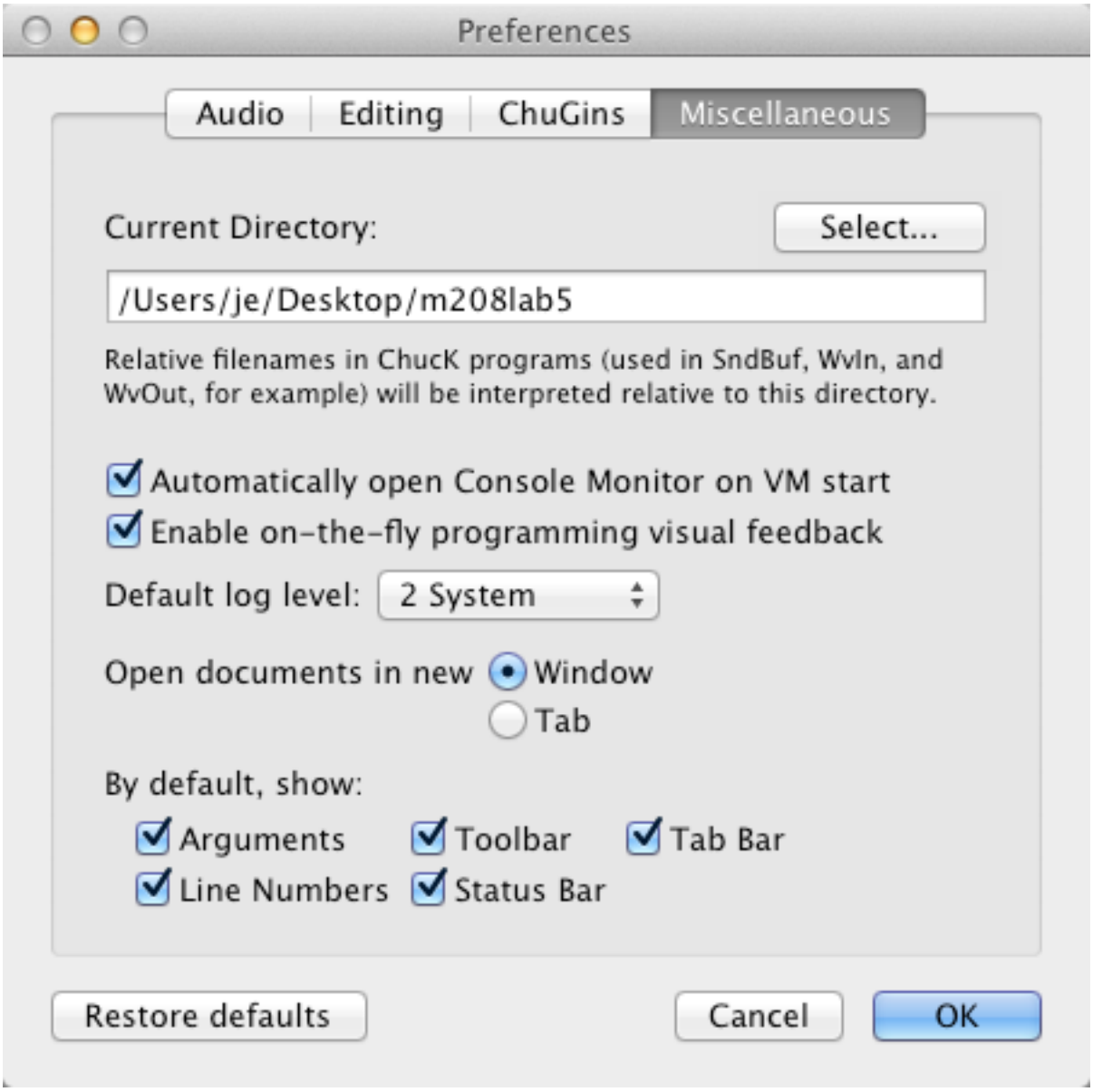
WIN
http://acad.carleton.edu/courses/musc208-00-w14/MIDIsetupWin.html

Lab 5 Keyboard Mapping, Blackhole, And Wavetables

Download and unzip m208lab5.zip to create the m208lab5 folder.

Open miniAudicle

Open Preferences and set the working directory to the m208lab5 folder.



Open keyboardPlayTemplate.ck

Open keyboardPlayTemplate.ck in miniAudicale.

Save As

Save as animalSounds.ck

Animal Sounds

The animals folder in the m208Lab5 folder contains several free animal sound samples from http://www.animal-sounds.org/farm-animal-sounds.html

Here's the Plan

You'll play them by typing.
Then you'll layer them.
Then you'll record them.
The you'll record them again with the blackhole object.
Then you'll add the shift key to play different sounds.
Then you'll learn about SndBuf arrays and overloaded functions.
The you'll record them into Audacity on your laptop.

Map Three Animal Sounds To The Keyboard Letters d j k

Choose three animal sounds and map them to the d j k keyboard keys. You could have used other keys, but the following code uses d j k.

In the code below the dog sound was assigned to d. Assign two other sounds to j and k.

```
206     else if ( msg.ascii == ascii_d )
207     {
208         playIt( "dog.wav" ); // animal sound name
209     }
210     else if ( msg.ascii == ascii_f )
211     {
212         playIt( "g" );
213     }
214     else if ( msg.ascii == ascii_g )
215     {
216         playIt( "g" );
217     }
218     else if ( msg.ascii == ascii_h )
219     {
220         playIt( "h" );
221     }
222     else if ( msg.ascii == ascii_j )
223     {
224         playIt( "j" ); // animal sound name
225     }
226     else if ( msg.ascii == ascii_k )
227     {
228         playIt( "k" );  // animal sound name
229     }
```

Create the SndBuf object

```
67 SndBuf buf => dac;
```

Modify The playIt() Function

This will be the same function we used in Lab 4 for the ABC song.

```
68
69 function void playIt( string str )
70 {
71     <<< "You typed", str >>>;
72     // samples are in the animals folder
73     "animals/" + str => buf.read;
74     0 => buf.pos;
75     buf.length() => now;
76 }
77
```

Run

Remember to deactivate the main miniAudicle window (click in the Console Monitor window) so your typing will not be entered into the code. Type d j k in any order. It should work.

Problem: You Can't Play Two Sounds At Once

Type d j k at the same time. You'll hear the sounds, but they'll be played one at a time, not all together.

Maybe we can fix it by creating the SndBuf object inside the playIt function. Cut the SndBuf buf => dac; line and paste it inside the playIt function.

```
68 function void playIt( string str )
69 {
70     <<< "You typed", str >>>;
71     SndBuf buf => dac;
72     // samples are in the animals folder
73     "animals/" + str => buf.read;
74     0 => buf.pos;
75     buf.length() => now;
76 }
77
```

Can you get d j k to play simultaneously? Not yet.

Multiple SndBuf Objects

Maybe we can fix it by creating a different SndBuf object for each sound. Let's try. Put the SndBuf line outside the function and add two more SndBuf objects.

```
67 SndBuf buf1 => dac;
68 SndBuf buf2 => dac;
69 SndBuf buf3 => dac;
70
71 function void playIt( string str )
```

The playIt function needs to know which SndBuf to play. We can do that with an if else block.

```
71 function void playIt( string str )
72 {
73     <<< "You typed", str >>>;
74     // samples are in the animals folder
75
76     if ( str == "dog.wav" )
77     {
78         "animals/" + str => buf1.read;
79         0 => buf1.pos;
80         buf1.length() => now;
81     }
82     else if ( str == "cat.wav" )
83     {
84         "animals/" + str => buf2.read;
85         0 => buf2.pos;
86         buf2.length() => now;
87     }
88     else if ( str == "duck.wav" )
89     {
90         "animals/" + str => buf3.read;
91         0 => buf3.pos;
92         buf3.length() => now;
93     }
94
95 }
96
```

Run

Type d j k. It works, but still only one sound at a time.

spork ~

Maybe we can fix it by sporking the three playIt() functions.

```
else if ( msg.ascii == ascii_d )
{
    spork ~ playIt( "dog.wav" ); // animal sound 1 name
}
```

Do the same for other two sounds.

Run And Type

Layered sounds now work. You can type d j k simultaneously.

Let's Record It

In Lab 4 we recorded the ABC song to ChucK's WvOut object. Maybe this will work.

```
SndBuf buf1 => WvOut w => dac;
SndBuf buf2 => WvOut w => dac;
SndBuf buf3 => WvOut w => dac;
"animalSong.wav" => w.wavFilename;
```

Nope! Console Monitor reports an error: "[animalSounds.ck]:line(68): 'w' has already been defined in the same scope..."

Let's try this.

```
SndBuf buf1 => WvOut w => dac;
SndBuf buf2 => w => dac;
SndBuf buf3 => w => dac;
"animalSong.wav" => w.wavFilename;
```

Run

Type d j k singly and simultaneously. Remember to click the Stop Virtual Machine button in the Virtual Machine window when you're done recording. Open the animalSong.wav in Audacity and see if it worked. It should. Close the Audacity window.

ChucK's Blackhole Object

ChucK's blackhole object is similar to dac in that it collects samples from any objects to its left. ChucK documents say it sucks samples. Unlike dac, blackhole does not play them.

Here's what http://wiki.cs.princeton.edu/index.php/ChucK/Record says about recording recording ChucK audio.

"the silent sample sucker strikes again

as in rec.ck, one patch to write to file is:

```
dac => gain g => WvOut w => blackhole;
```

the blackhole drives the WvOut, which in turns sucks samples from gain and then the dac. The WvOut can also be placed before the dac:

```
noise n => WvOut w => dac;
```

The WvOut writes to file, and also pass through the incoming samples."

You can use the blackhole object like this.

```
SndBuf buf1 => dac;
SndBuf buf2 => dac;
SndBuf buf3 => dac;
dac => WvOut w => blackhole;
"animalSong.wav" => w.wavFilename;
```

Run

Type d j k singly and simultaneously. Remember to click the Stop Virtual Machine button in the Virtual Machine window when you're done recording. Open the animalSong.wav in Audacity and see if it worked. It should. Close the Audacity window.

Try This

```
SndBuf buf1 => Gain g1 => WvOut w => dac;
SndBuf buf2 => Gain g2 => w => dac;
SndBuf buf3 => Gain g3 => w => dac;
dac => blackhole;
"animalSong.wav" => w.wavFilename;
1.0 => g1.gain;
0.25 => g2.gain;
0.1 => g3.gain;
```

Run

Type d j k singly and simultaneously. Remember to click the Stop Virtual Machine button in the Virtual Machine window when you're done recording. Open the animalSong.wav in Audacity and see if it worked. It should. Close the Audacity window.

Try This

Gains have changed, dac has been removed.

```
SndBuf buf1 => Gain g1 => WvOut w;
SndBuf buf2 => Gain g2 => w;
SndBuf buf3 => Gain g3 => w;
w => blackhole;
"animalSong.wav" => w.wavFilename;
.01 => g1.gain;
1.0 => g2.gain;
.3 => g3.gain;
```

Run

Type d j k singly and simultaneously AS FAST AS YOU CAN. Wait for the shreds to expire in the Virtual Machine window and then click the Stop Virtual Machine button. Open the animalSong.wav in Audacity and see if it worked. It should. Close the Audacity window.

The Shift Key

Let's use the Shift key to map the keys capital D J K to three more sounds.

kb.ck

Remember kb.ck? It's available online with many more ChucK examples (including rec.ck) at: http://chuck.cs.princeton.edu/doc/examples/. Go to that web page, search for kb, and open the kb link. Copy the kb.ck source code and paste it into a new miniAudicle document.

kb.ck reports three messages: msg.which, msg,key, and msg.ascii.

```
// check for action type
if( msg.isButtonDown() )
{
    <<< "down:", msg.which, "(code)", msg.key, "(usb key)", msg.ascii, "(ascii)" >>>;
}
```

You've already used msg.ascii to respond to letters and punctuation symbols. The reason we didn't use msg.which, or msg.key is because msg.which reports different values on different Windows and Mac keyboards, while msg.ascii stays the same.

Run kb.ck

Type d, hold down the shift key and type d again. You should see this in the Console Monitor window.

```
keyboard 'Keyboard' ready
down: 7 (code) 7 (usb key) 68 (ascii)       type d
down: 225 (code) 225 (usb key) 0 (ascii)    hold down shift key
down: 7 (code) 7 (usb key) 68 (ascii)       type d again
[chuck](VM): removing all (1) shreds...
```

Msg.ascii is zero for the Shift key, but msg.which is 225. Modifier keys report msg.ascii as zero, but they report different values for msg.which.

Users expect upper case D when the shift key is down and lower case d when the shift key is up. We want lower case d to be the dog sound, and want upper case D to be a different sound.

Try This

```
else if ( msg.ascii == ascii_d )
{
    if (msg.which == 225) // Shift key
        spork ~ playIt( "pony.wav" ); // animal sound 1
name
    else
        spork ~ playIt( "dog.wav" ); // animal sound 1 name
}
```

Type d followed by D. Does it work? Not yet!

Maybe we can declare a variable called shiftDown outside the while loop and then change its value in while loop when the shift key is pressed.

```
// infinite event loop
0 => int shiftDown;
while( true )
{
    // wait on event
    hi => now;

    // get one or more messages
    while( hi.recv( msg ) )
    {
        // check for action type
        if( msg.isButtonDown() )
        {
            if ( msg.which == 225 ) // shift down
            {
                1 => shiftDown;
            }
            // Row 1 2 3 4 5
            else if ( msg.ascii == ascii_1 ) // remove the
```

Then change this

```
else if ( msg.ascii == ascii_d )
{
    if ( shiftDown ) // Shift key
        spork ~ playIt( "pony.wav" ); // animal sound name
    else
        spork ~ playIt( "dog.wav" ); // animal sound name
}
```

Run

Type d, then D. Does it work? If not, maybe you forgot to add a new block to the playIt() function.

Now type lower case d again. You'd think you'd hear the dog but you still hear the pony. The problem is that shiftDown was set to one but was never reset to zero.

Maybe this will work.

```
else if ( msg.ascii == ascii_d )
{
    if ( shiftDown )
    {
        spork ~ playIt( "pony.wav" ); // animal sound 1 name
        0 => shiftDown;
    }
    else
        spork ~ playIt( "dog.wav" ); // animal sound 1 name
}
```

It seems to work, but it's still not quite right.

Type d. Press and release the Shift key and type d again. You'd expect to hear dog but you don't. The Shift key was set to one when the pressed down but was never set back to zero when the Shift key came up.

Make These Changes In The Code

Revert the msg.ascii == ascii_d block to:

```
else if ( msg.ascii == ascii_d )
{
    if ( shiftDown )
        spork ~ playIt( "pony.wav" ); // animal sound 1 name
    else
        spork ~ playIt( "dog.wav" ); // animal sound 1 name
}
```

Reset shiftDown to zero in the msg.isButtonUp block.

```
else if( msg.isButtonUp() )
{
    if ( msg.which == 225 ) // shift down
    {
        0 => shiftDown;
    }

}
```

Now it should run as expected.

Add More Sounds

Add two more shiftDown sounds for a total of six; three with the shift key down and three with the shift key up.

SndBuf Arrays

This method becomes unwieldy when we add more sounds and SndBufs. Creating an array of SndBuf objects can simplify the code.

```
// create an array of six SndBuf's
// arrays are indexed starting from 0
// buf[0] is the first SndBuf and used to be buf1;
// buf[5] is the last of the 6 SndBuf's
SndBuf buf[6];

// connect them to the dac
// buf.cap() returns the number of elements in the array
// right now we know it's 6 but if we decide to change it
// buf.cap() will know as well
for ( 0 => int ix; ix < buf.cap(); ix++ )
{
    buf[ix] => dac;
}

// we only need 0ne WvOut;
dac => WvOut w => blackhole;
"animalSong.wav" => w.wavFilename;
```

Multiple playIt() Methods

ChucK allows you to have more than one function with the same name as long as the parameter types are different. The term for this in computer science is an "overloaded" function.

We'll use two playIt methods. One for the animal sound key presses and one for all other key presses. The first function, playIt(string wavFilename), has as single string parameter that is the name of the wav file. The second function, playIt(string wavFilename, int bufIndex), has two parameters: a string for the name of the wave file, and an integer representing which SndBuf to use.

Add a second playIt function to your code.

```
// this playIt function is used for all non animal keys
function void playIt( string str )
{
    <<< "You typed", str >>>;
    // samples are in the animals folder
}

// this playIt function is only used for animal keys
// ix is the SndBuf array index
function void playIt( string str, int ix )
{
    <<< "You typed", str >>>;
    // samples are in the animals folder

    "animals/" + str => buf[ix].read;
    0 => buf[ix].pos;
    buf[ix].length() => now;
 }
```

Modify The While Loop

Sporking the playIt function allows the sounds to run in parallel "shreds in real time.

```
else if ( msg.ascii == ascii_d )

{
    if ( shiftDown )
        spork ~ playIt( "pony.wav", 0 ); // animal sound
name
    else
        spork ~ playIt( "dog.wav", 1 ); // animal sound
name
}
else if ( msg.ascii == ascii_f )
{
    playIt( "g" ); // non animal sound
}
```

and later

```
else if ( msg.ascii == ascii_j )
{
     if ( shiftDown )
        spork ~ playIt( "pig.wav", 2 ); // animal sound
name
    else
        spork ~ playIt( "duck.wav", 3 ); // animal sound
name
}
else if ( msg.ascii == ascii_k )
{
    if ( shiftDown )
        spork ~ playIt( "sheep.wav", 4 ); // animal sound
name
    else
        spork ~ playIt( "cat.wav", 5 ); // animal sound
name
}
```

The Animals Stampede

Type d j k D J K as fast as you can.

Does it work? It should.

Record it into Audacity on Your Laptop

Remove any WvOut or blackhole statements from your code.

```
47  => int ascii_forwardSlash; // forwardSlash

SndBuf buf1 => dac;
SndBuf buf2 => dac;
SndBuf buf3 => dac;

function void playIt( string str )
```

Mac: You need Audacity, Au Lab, and Soundflower installed on your laptop.

Win: You need Audacity version 2.05 and may need updated sound drivers.

End of Lab 5.