Songlib: quickstart

Song Li Buser

Revision Date: January 22, 2015

Making your first song

This page assumes you have properly installed **songlib**and the initial samples. If you haven't yet, please follow the directions in install .

The easiest way to write a **songlib**program is to base it upon an existing program. Here is a working program that you can compile and, in the future, modify. These files are found in the quikstart directory but are repeated here:

Name this file *quickstart.c*:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <songlib/util.h>
#include <songlib/songlib.h>

#define dir "/usr/local/share/samples/mandolin/"
#define base "sample_"

/* change PROGRAM_NAME and PROGRAM_VERSION appropriately */

char *PROGRAM_NAME = "quickstart";
char *PROGRAM_VERSION = "0.01";

static int last = 0;

static void
verse(int instrument,int octave)
    {
    startMeasure();

    //line 1

    chord(H,instrument,octave+1,C,-5,-12,(int) 0);
    chord(Q,instrument,octave+1,C,2,-5,-12,(int) 0);
    chord(Q,instrument,octave,C,7,14,(int) 0);

    chord(Q,instrument,octave+1,E,-9,-16,(int) 0);
    chord(H,instrument,octave,D,5,18,(int) 0);
    chord(Q,instrument,octave+1,F,-8,-15,(int) 0);

    chord(Q,instrument,octave+1,F,-8,-15,(int) 0);
```

```
chord(H,instrument,octave,D,7,18,(int) 0);
chord(Q,instrument,octave+1,F,-8,-15,(int) 0);

chord(Q,instrument,octave+1,E,-9,-16,(int) 0);
chord(Q,instrument,octave,D,5,-2,(int) 0);
chord(H,instrument,octave+1,C,-5,-12,(int) 0);

//line 2

play(Q,instrument,octave,C);
min(H,instrument,octave,G);
play(H,instrument,octave,G);
play(Q,instrument,octave,A);
play(Q,instrument,octave,G);
play(Q,instrument,octave,F);

play(Hd,instrument,octave,G);
chord(Q,instrument,octave+1,E,-9,-16,(int) 0);

chord(Q,instrument,octave,C,7,16,(int) 0);
chord(Q,instrument,octave+1,E,-9,-16,(int) 0);
chord(Q,instrument,octave,D,-2,(int) 0);
chord(Q,instrument,octave,C,7,12,(int) 0);

chord(H,instrument,octave+1,C,-8,-12,(int) 0); //first half of next measure

//line 3

play(H,instrument,octave,C); //last half of measure

play(Q,instrument,octave,G);
play(Q,instrument,octave,F);
play(H,instrument,octave,E);

play(H,instrument,octave,F);
chord(H,instrument,octave,G,-7,(int) 0);

chord(H,instrument,octave,D,7,(int) 0);
chord(Hd,instrument,octave,D,-3,(int) 0); //first quarter of next measure

//line 4

play(Q,instrument,octave,C); //last 3/4 of measure
play(Q,instrument,octave,C);
play(Q,instrument,octave,D);

play(Q,instrument,octave,E);
play(Q,instrument,octave,F);
play(Q,instrument,octave,E);
play(Q,instrument,octave,D);

play(Q,instrument,octave,C);
play(Q,instrument,1,A);
chord(H,instrument,octave,F,-3,-7,(int) 0);
```

```
    chord(Q,instrument,octave,F,-3,-7,(int) 0);

    if (last) setSustain(0.999999);

    chord(Hd,instrument,octave,C,4,7,IX);

    checkMeasure();
    }

int
main()
    {
    int instrument;
    int octave = 2;

    songInit();

    instrument = readScale(dir,base);

    setTempo(250);
    setTime(4,4);
    setStride(0.05);
    setSustain(0.99995);
    setAmplitude(0.4);

    openOutput("quickstart.rra",0,0);

    verse(instrument,octave);
    last = 1;
    verse(instrument,octave);

    closeOutput();

    return 0;
    }
```

Next, add a file named *makefile* to your working directory with the following content:

```
INCLUDE=/usr/local/include/songlib/
LIB=/usr/local/lib/

# Place the names of all your tracks here (as RRA filenames)

RRAS = quickstart.rra

all : $(RRAS)

# comment out the rplay line if you don't want automatic playing

%.rra   : %.x
        ./$<

%.x     : %.c
```

```
gcc -Wall -g -o $@ -I$(INCLUDE) $< -L$(LIB) -lsong -lm

play    : $(RRAS)
        rrafastmixer -a0.5 $(RRAS) | rplay

.PHONY : clean

clean :
        -rm -f $(RRAS)
```

IMPORTANT: Make sure all indented lines in your makefile begin with a tab character and not spaces.

Finally, type:

```
make
```

and a **songlib** program should be built and run, resulting in an RRA audio file named:

```
quickstart.rra
```

The RRA can be played with this command:

```
rplay quickstart.rra
```

or this command:

```
make play
```

The latter plays the audio file at a reduced volume.

Decomposing quickstart.c

For this discussion, focus your attention on the *main* function of the *quickstart.c* file.

Initializing songlib

The *songInit* function readies the **songlib** system for audio production. It must be the first **songlib** library function called and is required.

Reading instrument notes

The *readScale* function is used to read in a set of notes in a given directory. For example, the call:

```
readScale("./","bright_");
```

would read in all the RRA files with prefix *bright_* in the current directory. These RRA files are then attached to the instrument number that *readScale* returns. For more information on the naming of *RRA* files that can be read in with *readScale*, see installing samples. For more information on *readScale* and other methods for reading in RRA files, see reading samples.

Setting parameters

Various settings of **songlib** are controlled with function such as:

```
setTempo(250);
setTime(4,4);
setStride(0.05);
setSustain(0.99995);
setAmplitude(0.4);
```

For more information on these and other similar functions, see controlling output and keeping time.

Designating the output

Songlib is capable of working with multiple sample rates and sample sizes. Unless told otherwise, **songlib**, however, assumes a sample rate of 44100 samples per second and a sample size of 16 bits per sample. One changes the sample rate and size globally with the *openOutput* function. This ensures the output file will be written with the given sample rate and size. As a side effect, if **songlib** cannot figure out the sample rate and size for a particular operation, these given values will be used.

If the call to *openOutput* is passed zeroes for the sample rate and size, as in:

```
openOutput("track.rra",0,0);
```

then the default values of 44100 samples per second and 16 bits per sample are used. To set the output rra to 96000 samples per second and 32 bits per sample, for example, the call would be:

```
openOutput("track.rra",96000,32);
```

Playing notes

Between the calls to *openOutput* and *closeOutput*, one makes calls to the various note playing functions in **songlib**. The example program seen previously makes a number of these calls, primarily to *play* and *chord*. For more information on these note playing functions, see note playing.

Writing the audio file

The last **songlib** function to be called is *closeOuput*. This function does the actual writing of the RRA file specified in the call to *openOutput*. This function is required.