3D Printed Record

by amandaghassaei

https://www.youtube.com/embed/NM7hwAuXqCE

In order to explore the current limits of 3D printing technology, I've created a technique for converting digital audio files into 3D-printable, 33rpm records and printed a few functional prototypes that play on ordinary record players. Though the audio quality is low -the records have a sampling rate of 11kHz (a quarter of typical mp3 audio) and 5-6 bit resolution (less than one thousandth of typical 16 bit resolution)- the songs are still easily recognizable, watch the video above to see the process and hear what the records sound like. Also check out my laser cut records, made on wood, paper, and acrylic.

https://www.youtube.com/embed/Anaau7lh9iU

This past year I've been posting a lot of audio projects, specifically, I've been experimenting with using relatively simple tools and techniques and very little memory to approximate and recreate digital audio signals. A great example is my Arduino Vocal Effects Box, where I used an Arduino to perform realtime pitch-bending on an incoming audio signal. Through these projects, I've learned that audio is a very resilient medium, it can take a fair amount of abuse (in the form of distortion and compression) while still maintaining most of the integrity of the original sound. The key is as long as you loosely approximate the overall shape of an audio signal, the output will sound reasonably recognizable. We have evolution to thank for this: as we hear audio, some complicated processing goes on in our brains that makes us very good at ignoring noise and focusing on the important pieces of information coming through. We can work off of relatively few cues (sometimes these even include contextual or visual cues) to piece together mangled or noisy audio and make sense of it; this is how we are able to focus on one voice in crowded room or decipher a message sent over a cheap walkie talkie.

This project was my first experiment extending this idea beyond electronics. I printed these records on a UV-cured resin printer called the Objet Connex500. Like most 3D printers, the Objet creates an object by depositing material layer by layer until the final form is achieved. This printer has incredibly high resolution: 600dpi in the x and y axes and 16 microns in the z axis, some of the highest resolution possible with 3D printing at the moment. Despite all its precision, the Objet is still at least an order of magnitude or two away from the resolution of a real vinyl record. When I first started this project, I wasn't sure that the resolution of the Objet would be enough to reproduce audio, but I hoped that I might produce something recognizable by approximating the groove shape as accurately as possible with the tools I had.

In this Instructable, I'll demonstrate how I developed a workflow that can convert any audio file, of virtually any format, into a 3D model of a record, and how I optimized these records for playback on a real turntable. The 3D modeling in this project was far too complex for traditional drafting-style CAD techniques, so I wrote an program to do this conversion automatically. It works by importing raw audio data, performing some calculations to generate the geometry of a record, and eventually exporting this geometry straight to a 3D printable file format. Most of the heavy lifting is done by Processing, an open source programming environment that's often used for 2D and 3D graphics and modeling applications.

Here's a basic overview of my Processing algorithm:

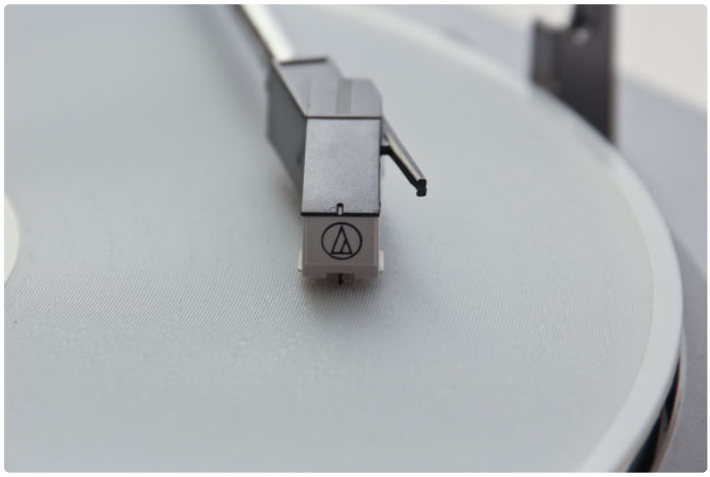
use raw audio data to set the groove depth- parse through the raw audio data, this is the set of numbers that defines the shape of the audio waveform, and use this information to set the height of the bottom of a spiral groove. This way, when a turntable stylus moves along the groove it will move vertically in the same path as the original waveform and recreate the original audio signal.
draw record and groove geometry- A 3D model is essentially a list of triangles arranged in 3D space to create a continuous mesh, use the data from the last step and some general record parameters (record diameter, thickness, groove width, etc) to generate the list of triangular faces that describes the record's shape and the detailed spiral groove inscribed on its surface.
export model in STL format- the STL file format is understood by all 3D printers, export the geometry calculated in the last step as an STL file. To get Processing to export straight to STL, I used the Model Builder Library written by Marius Watz (if you are into Arduino/Processing and 3D printing I highly recommend checking this out, it works great).

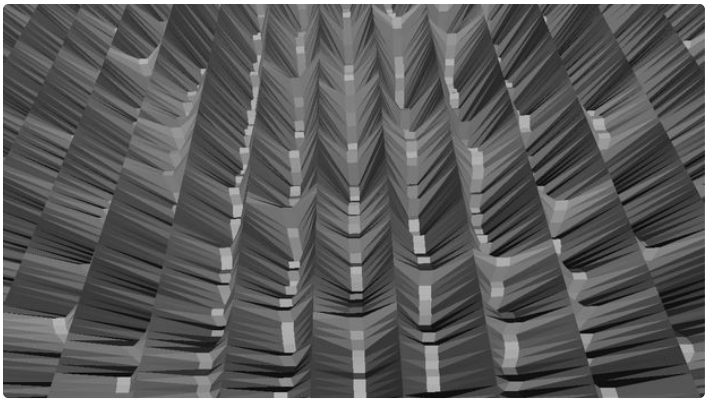
I've uploaded some of my complete record models to the 123D gallery as well as the Pirate Bay. Check Step 6 for a complete listing of what's there and what I plan on posting. Alternatively, you can go to Step 7 to download my code and learn how to make printable record models from your own audio.

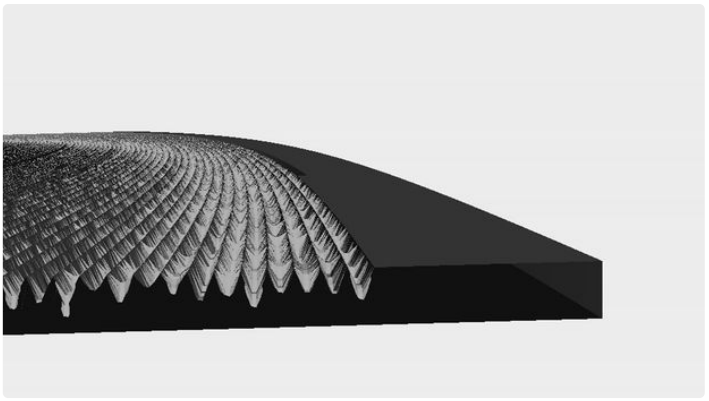
Special thanks to Randy Sarafan, Steve Delaire, Arthur Harsuvanakit, Phil Seaton, and Audrey Love for their help with this project.

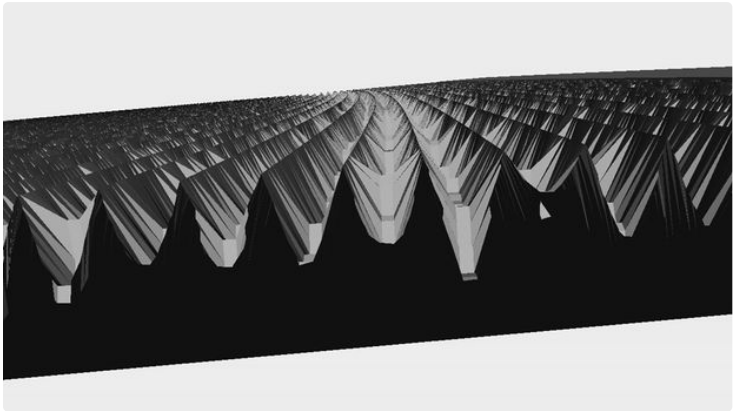
Here's another video that gives a great overview of the printing process and shows the printers at work:

//www.youtube.com/embed/lQi8FUsZ8OY









Step 1: How Does a Record Work?

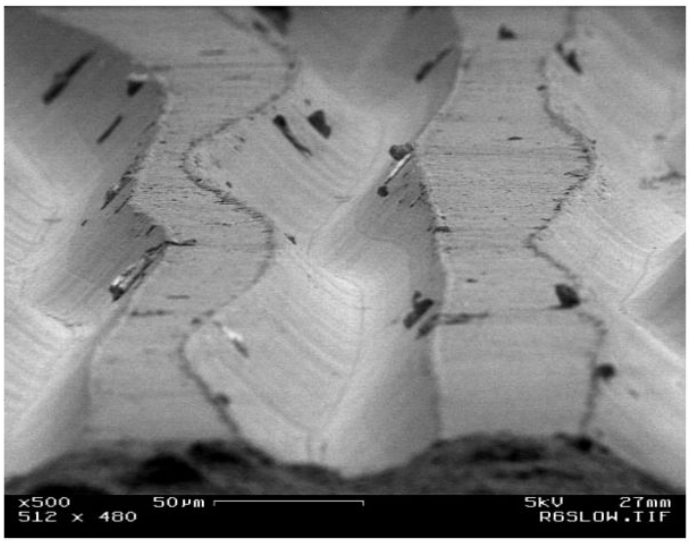
The basic mechanism of a record player is very simple. The record moves at a constant rotational speed (usually 33.3 or 45 rpm) and a needle (also called a stylus) moves along a long spiral groove cut into the record's surface. As the record spins, the needle hits tiny bumps in the groove and vibrates to produce audio signals. I won't get into the specifics of how the needle extracts data from the record, but it is *really* interesting and there's a great demo of it here.

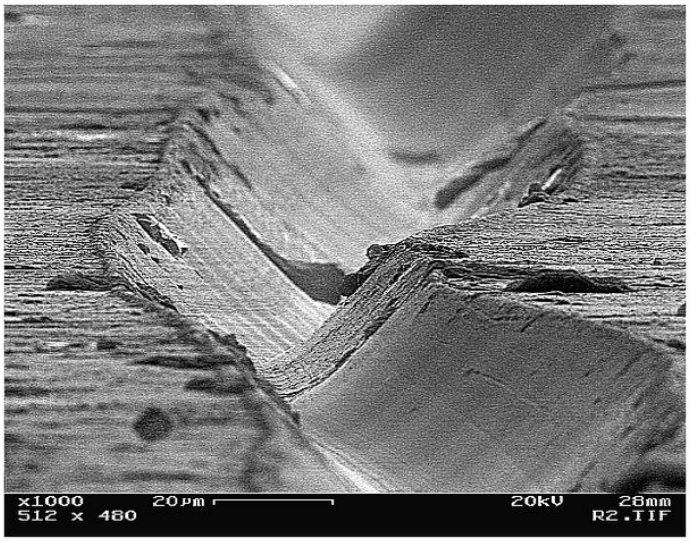
The record player and record cutter were invented by Edison in 1877. Due to a lack of precise machinery and technique at the time, the grooves on the first records were much larger than those on modern microgroove records and, subsequently, the audio signals were much noisier. This is a similar situation that I found myself in when starting this project: despite the high precision of the Objet machines, the resolution is nowhere near modern vinyl quality. Here and here are two examples of Edison's first phonograph tests. You can hear that the quality of recording of these tests is pretty close to what I've been able to 3d print; although I can't find the exact specs on these records, I'd imagine that the scale of the grooves was similar to what I was working with.

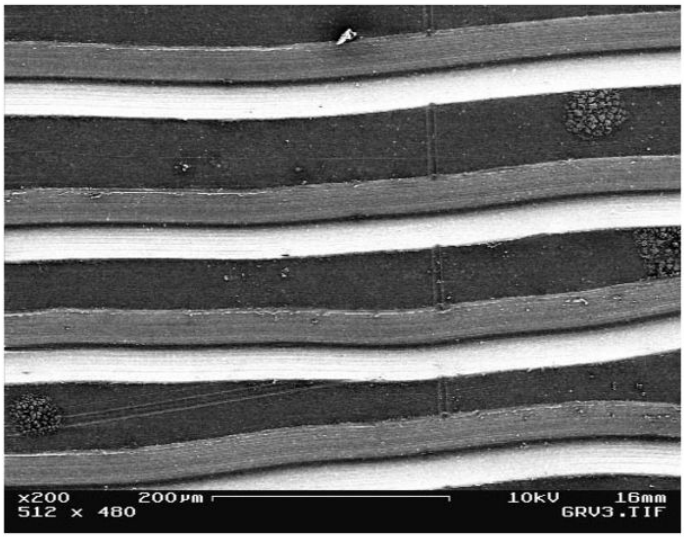
To give you an idea of the resolution of a modern record, check out the images above. Figs 1-3 are from Chris Supranowitz, a researcher at The Institute of Optics at the University of Rochester. These are close up images of a vinyl record, taken with an electron microscope. The dark objects in figs 1 and 2 are tiny particles of dust. Fig 3 is a bird's eye view of the record grooves, the darker regions are the top (uncut) surface of the record.

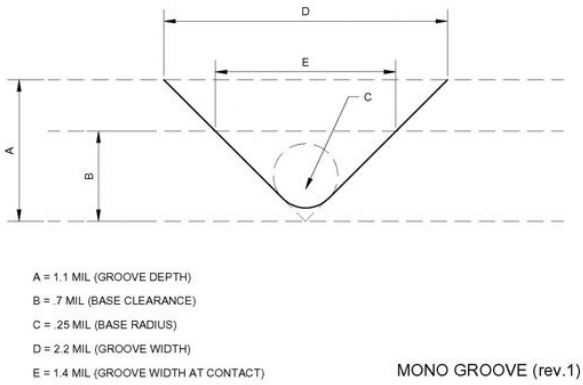
Fig 4 was made by branku62 at vinylengine.com, it shows the profile dimensions of a standard microgrove mono groove, this is what you would find on a modern mono 33 or 45 (stereo grooves are actually cut a bit smaller). In the diagram 1 mil = 1/1000", which is about 25um. Microgroove records require a stylus with a 0.7 to 1.0 mil radius tip, the tip makes contact with the groove at E in fig 1, a width of about 1.4 mil. The total depth of the groove is around 1.1 mil. These dimensions match up nicely with the dimensions of the electron microscope images.

Fig 5 is from Ron Geesin and Mark Berresford's webs ite, it shows the groove depths of the older 78's. These records were much more coarse than microgroove records, both the needle and grooves were about 3x as large in every dimension. Fig 2 shows the groove depth for 78's was somewhere between 2.2 and 3.6 mil. The stylus radius was around 2.7 mil.









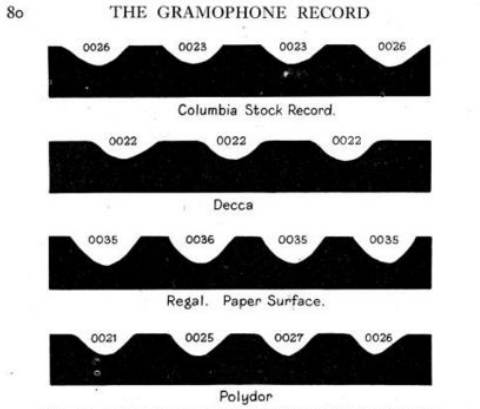


FIG. 36.—PHOTOMICROGRAPHS OF SECTIONS OF VARIOUS RECORDS
Showing the variation in dimensions and shapes of the grooves. The figures show the depths of the grooves in decimal parts of an inch.

Step 2: Printer Specs

Here at Instructables HQ, we have access to Autodesk's fleet of Objet Connex 500 printers. These printers use UV light to cure resin layer by layer until a complete model is produced. They are very different from the fused deposition printers you may have seen or used before (MakerBot, RepRap, Up!, etc), not only can they print out of many types of materials (ranging from flexible rubbery material to hard polymer), but they are also extremely precise. In the x and y axes they have 600dpi resolution (that's about 42microns), and in the z axis they have a resolution of 16microns.

Before I started printing anything, I used these numbers to calculate the resolution I'd be able to achieve- so I could decide if this project was even worth pursuing any further. First I wanted to make sure that I would be able to get a good sampling rate on my audio. Sampling rate is the amount of samples per second in a song. Usually the sampling rate is 44.1kHz (or 44,100 samples a second). When the sampling rate drops below about 40kHz the higher frequencies of a song start losing their detail, but depending on the song you can go down to 10kHz sampling rate without too much of a problem.

To calculate the sampling rate of the 3D printed record I used the following relationship:

sampling frequency = (resolution per inch)* (inches per revolution)*(revolutions per second)
in order to maximize the sampling frequency, I want all of these numbers (res/inch, inch/rev, rev/sec) to be as high as possible

First I'll start with revolutions per second. Record players typically play at two different speeds: 33.3 and 45rpm. (Some record players also have a 78rpm speed, but this is less common and only used for very old records). I wanted to use the lower 33.3RPM speed in order to make this more like a real 12" record (45 RPM is only used for 7" records, and 33RPM for the full sized 12") and so that I could fit more audio onto each side of the disc.

revolutions per second = (revolutions per

minute)/(seconds per minute)
revolutions per second at 33 rpm = 33.3/60 = 0.55

Next is inches per revolution, this number depends on the circumference of the disk where the needle is hitting it. The largest sized records are 12" in diameter (30cm). According to the RIAA standards, the outermost groove of a 12" record falls at a radius of 5.75" and the innermost groove falls at about 2.25". I'll use these numbers to determine the range of sampling rates I can achieve at 33RPM. The circumference (the distance in inches traveled by the needle during one revolution of the record) is calculated as follows:

inches per revolution = 2*pi*(radius of needle)
max inches per revolution = 2*pi*5.75 =~ 36
min inches per revolution = 2*pi*2.35 =~ 15

I already know that the resolution per inch of the 3D printer is 600 (600 dpi in the x and y axes). So combining this all I get:

sampling frequency = (resolution per inch)* (inches per revolution)*(revolutions per second)
max sampling frequency at 33 rpm = 600*36*0.55 =~ 12000 = 12kHz
min sampling frequency at 33 rpm = 600*15*0.55 =~ 4900 = 4.9kHz

This is a pretty good starting point. If I scale this to 45rpm instead of 33 the sampling rate becomes:

max sampling frequency at 45 rpm = 600*36*0.75 =~ 16000 = 16kHz
min sampling frequency at 45 rpm = 600*15*0.75 =~ 6700 = 6.7kHz

I'll keep this option in mind in case sampling rate becomes an issue. The other piece of information that I needed was the bit depth I'd be able to achieve with the Objet printer. Bit depth is the resolution of the audio data. Most audio these days in 16 bit, meaning each sample can have one of 65536 (2^16) possible values. 8 bit audio has only 256 (2^8) steps of resolution and still sounds pretty close to the original. Even going down to 3 and 4 bit sounds recognizable. (I should note here that the music commonly referred to as "8-bit" like the music in early Nintendo games is actually 1 bit resolution, it's called 8 bit because it was first made with 8 bit computers, not with 8 bit resolution).

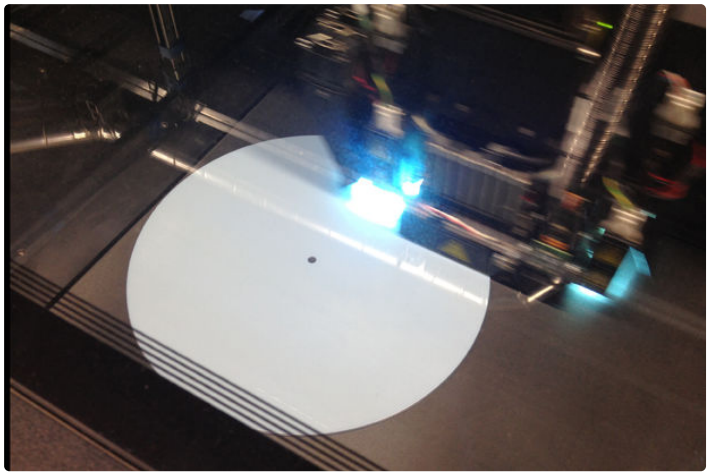
Since the z axis is the most precise axis on the Objet printer, I wanted to print my record so that the needle vibrates vertically in the groove to trace out the audio wave to maximize my bit depth. The following equation calculates the vertical distance that the needle will move as it traces the a wave of a given bit depth:

vertical displacement of needle = (2^bit depth)* (precision of z axis)
where the precision of the z axis is 16micron. I used this to calculate the following table:

bit depth vertical displacement steps of resolution

2 64um 4
3 128um 8
4 256um 16
5 512um 32
6 1.024mm 64
7 2.048mm 128
8 4.096mm 256

The bolded rows in the table are the numbers that I wanted to shoot for with this project. A vertical amplitude of 64-512um is an order of magnitude (~10x) larger than the amplitude of a vinyl record groove, but I felt like I'd probably be able to get away with it and still maintain a reasonable bit depth.



Step 3: First Tests

First I prepared some test files to print to get an idea of what is possible with the printer and optimize the dimensions of the grooves. These record files have circular grooves on them containing sine waves of various frequencies, amplitudes, groove depths, groove widths, and beveled groove edges. (When I say that the groove "contained" a sine wave, I mean the bottom of the groves moves up and down in a sinusoidal pattern around the record). I generated all of these files in Processing using the ModelBuilder library to export straight to STL.

TEST ONE:

My first test record had 72 grooves on it, screen shots of the model are shown in figs 2 through 6 I tested two frequencies of sine waves:

1000 cycles per revolution = 555Hz at 33RPM
500 cycles per revolution = 277Hz at 33RPM

I tested a few different amplitudes, depths, and groove widths for these frequencies and gave each groove a constant bevel size of 2px on each side (you can see in fig 5 how the edges of the groove flare outward). I printed the record in Objet's Vero Clear material, this material is a fairly hard, clear resin. I printed the file with the "smooth" setting to prevent any support material from being deposited in the grooves. Unfortunately, when I was ready to make this print we were having some problems with power in our shop, so I had to use another Objet machine that was not set up for high resolution printing; the best I could do was 300DPI X/Y resolution with 30um Z steps. This is half the resolution that each of these axes is capable of, meaning the print came out at $(1/2)^3$, or 1/8th resolution overall. The results are shown in the video below (the grooves were not deep enough to keep the needle inside, so I had to hold in it place with my hand). The record was also a little big for my record player, I decreased the diameter of my STL file to 11.8" in later versions.

https://www.youtube.com/embed/mQuOyFzDwrg

In this video you can hear a periodic frequency sweep on top of the steady sine wave (best heard w headphones). This sweeping sound is caused by the needle moving over the thousands of tiny parallel bumps in the print caused by adjacent print-heads on the Objet machine. This noise is unavoidable, but increasing the strength of the signal will help to make it less noticeable.

The Processing sketch that generated this record is given below:

```
<pre>//sine tests
//by Amanda Ghassaei
//Dec 2012
//https://www.instructables.com/id/3D-Printed-Record/

/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3 of the License, or
 * (at your option) any later version.
*/



import processing.opengl.*;
import unlekker.util.*;
import unlekker.modelbuilder.*;
import ec.util.*;

UVertexList recordPerimeterUpper,recordPerimeterLower,recordHoleUpper,recordHoleLower;//storage for perimeter and center hole of record
UVertexList lastEdge;//storage for conecting one groove to the next
UGeometry geo;//storage for stl geometry

//variables
float theta;//angle variable
float thetaIter = 10000;//how many values of theta per cycle
float radius;//variable to calculate radius of grooves
int diameter = 12;//diameter of record in inches
float innerHole = 0.286;//diameter of center hole in inches
float innerRad = 2.35;//radius of innermost groove in inches
float outerRad = 5.75;//radius of outermost groove in inches
float grooveSpacing = 20;//pixel spacing of grooves
float bevel = 2;//pixel width of groove bevel

//record parameters
float recordHeight = 0.08;//height of record in inches
int recordBottom = 0;//height of bottom of record

//parameters to test
float amplitude[] = {2,4,8};//in units of 16 micron steps (remember this is the amplitude of the sine wave, the total vert displacement will be twice this)
int frequency[] = {1000,500,0};//cycles per rotation
float depth[] = {0.5,1,0};//how many 16 microns steps below the surface of the record to print the uppermost point of the groove
float grooveWidth[] = {1,2,3};//in 600dpi pixels

float incrNum = TWO_PI/thetaIter;//calculcate inrementation amount

int grooveNum = 0;//variable for keeping track of how long this will take

void setup() {//everything that executes in this sketch is contained in the setup()

  geo = new UGeometry();//place to store geometery of verticies

  setUpVariables();//convert units, initialize etc
  setUpRecordShape();//draw basic shape of record
  drawGrooves();//draw in grooves

  geo.writeSTL(this, "test.stl");//write stl file from geomtery

}

void setUpVariables(){

  //convert everything to inches
```

```
  float micronsPerInch = 25400;//scalingfactor
  float dpi = 600;
  byte micronsPerLayer = 16;//microns per vertical print layer

  grooveSpacing /= dpi;
  bevel /= dpi;
  for(byte i=0;i<3;i++){
    amplitude[i] = amplitude[i]*micronsPerLayer/micronsPerInch;
    depth[i] = depth[i]*micronsPerLayer/micronsPerInch;
    grooveWidth[i] /= dpi;
  }

}

void setUpRecordShape(){

  //set up storage
  recordPerimeterUpper = new UVertexList();
  recordPerimeterLower = new UVertexList();
  recordHoleUpper = new UVertexList();
  recordHoleLower = new UVertexList();

  //get verticies
  for(theta=0;theta<TWO_PI;theta+=incrNum){
    //outer edge of record
    float perimeterX = diameter/2+diameter/2*cos(theta);
    float perimeterY = diameter/2+diameter/2*sin(theta);
    recordPerimeterUpper.add(perimeterX,perimeterY,recordHeight);
    recordPerimeterLower.add(perimeterX,perimeterY,recordBottom);
    //center hole
    float centerHoleX = diameter/2+innerHole/2*cos(theta);
    float centerHoleY = diameter/2+innerHole/2*sin(theta);
    recordHoleUpper.add(centerHoleX,centerHoleY,recordHeight);
    recordHoleLower.add(centerHoleX,centerHoleY,recordBottom);
  }

  //close vertex lists (closed loops)
  recordPerimeterUpper.close();
  recordPerimeterLower.close();
  recordHoleUpper.close();
  recordHoleLower.close();

  //connect verticies
  geo.quadStrip(recordHoleUpper,recordHoleLower);
  geo.quadStrip(recordHoleLower,recordPerimeterLower);
  geo.quadStrip(recordPerimeterLower,recordPerimeterUpper);

  //to start, outer edge of record is the last egde we need to connect to with the outmost groove
  lastEdge = new UVertexList();
  lastEdge.add(recordPerimeterUpper);

  println("record drawn, starting grooves");
  grooveNum = 0;//variable for keeping track of how much longer this will take

}

void drawGrooves(){

  UVertexList grooveOuterUpper,grooveOuterLower,grooveInnerUpper,grooveInnerLower;//groove verticies

  //set up storage
  grooveOuterUpper = new UVertexList();
  grooveOuterLower = new UVertexList();
  grooveInnerUpper = new UVertexList();
  grooveInnerLower = new UVertexList();

  //DRAW GROOVES
  radius = outerRad;//outermost radius (at 5.75") to start
  for(byte frequencyIndex=0;frequencyIndex<2;frequencyIndex++){
    for(byte amplitudeIndex=0;amplitudeIndex<3;amplitudeIndex++){
      for(byte grooveDepthIndex=0;grooveDepthIndex<2;grooveDepthIndex++){
        for(byte grooveWidthIndex=0;grooveWidthIndex<3;grooveWidthIndex++){
          for(byte copies=0;copies<2;copies++){

            //clear lists
```

```
          grooveOuterUpper.reset();
          grooveOuterLower.reset();
          grooveInnerUpper.reset();
          grooveInnerLower.reset();

          for(theta=0;theta<TWO_PI;theta+=incrNum){//for theta between 0 and 2pi

            float sineTheta = sin(theta);
            float cosineTheta = cos(theta);

            //calculate height of groove
            float grooveHeight = recordHeight-depth[grooveDepthIndex]-amplitude[amplitudeIndex]+amplitude[amplitudeIndex]*sin(theta*frequency[frequencyIndex]);

            grooveOuterUpper.add((diameter/2+(radius+bevel)*cosineTheta),(diameter/2+(radius+bevel)*sineTheta),recordHeight);
            grooveOuterLower.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);
            grooveInnerLower.add((diameter/2+(radius-grooveWidth[grooveWidthIndex])*cosineTheta),(diameter/2+(radius-grooveWidth[grooveWidthIndex])*sineTheta),groove
Height);
            grooveInnerUpper.add((diameter/2+(radius-grooveWidth[grooveWidthIndex]-bevel)*cosineTheta),(diameter/2+(radius-grooveWidth[grooveWidthIndex]-bevel)*sineT
heta),recordHeight);

          }

          //close vertex lists (closed loops)
          grooveOuterUpper.close();
          grooveOuterLower.close();
          grooveInnerUpper.close();
          grooveInnerLower.close();

          //connect verticies
          geo.quadStrip(lastEdge,grooveOuterUpper);
          geo.quadStrip(grooveOuterUpper,grooveOuterLower);
          geo.quadStrip(grooveOuterLower,grooveInnerLower);
          geo.quadStrip(grooveInnerLower,grooveInnerUpper);

          //set new last edge
          lastEdge.reset();//clear old data
          lastEdge.add(grooveInnerUpper);

          radius -= grooveSpacing+grooveWidth[grooveWidthIndex];//set next radius

          //tell me how much longer
          grooveNum++;
          print(grooveNum);
          println(" of 72 grooves drawn");
        }
        radius -= 2*grooveSpacing;//extra spacing
      }
      radius -= 2*grooveSpacing;//extra spacing
    }
    radius -= 2*grooveSpacing;//extra spacing
  }
  radius -= 2*grooveSpacing;//extra spacing
 }

 geo.quadStrip(lastEdge,recordHoleUpper);//close remaining space between last groove and center hole

}
```

TEST TWO:

In my next test I made a record with 108 grooves, still sine waves, but this time I made the grooves deeper, increase the bevel of each groove to equal half the amplitude of the sine wave, and tried out three different frequencies: 555hz, 277hz, and 139hz (1000, 500, and 250 cycles per revolution at 33.3rpm). I also tested different amplitudes (4, 8 and 16 steps), groove depths (2, and 3 steps below the top of the record), and groove widths (1, 2 and 3 pixels). Since our shop came back online, I switched printers and started printing with Objet's Vero White material, which is similar to Vero Clear in texture, but (as you might image) is a translucent white color. This time I was finally able to print with the full 16 micron and 600 dpi resolution of the printer. Here is a video of the results:

https://www.youtube.com/embed/qVDAOsswBiE

TEST THREE:

In my third test I increased the resolution of my stl file to test out some higher frequency sine waves. I used 22000 points per revolution to draw out the sine waves (as opposed to 10000 in my previous tests), this puts me at about the max resolution I can get with 600dpi (calculated in the last step). I tested three frequencies: 1110hz, 832hz, and 694hz (2000, 1500, and 1250 cycles per revolution at 33.3rpm). I also tested different amplitudes (12 and 16 steps) and groove widths (2 and 3 px). Here is the video:

https://www.youtube.com/embed/V-Lqj0wB7nY

RESULTS:

At the end of all these tests I learned a few things about 3d printing records with the Objet:

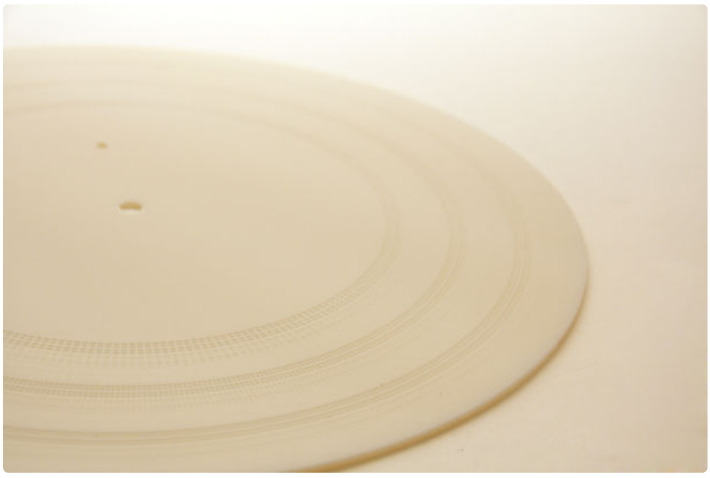
Groove Depth min of 48um below top of record - I found that grooves that kept the waveform at a minimum of 48um (or 3 16 micron steps) below the top of the record kept the needle in place while being played. This was true for all the frequencies I tested.

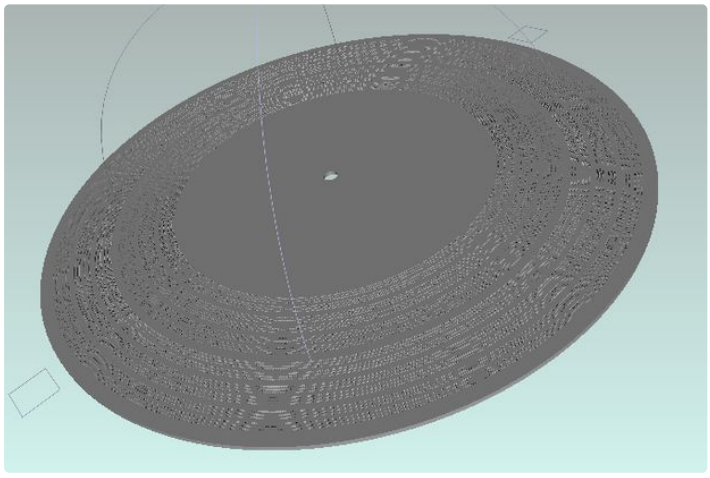
Groove Width 2px - At lower frequencies I found that the 2px grooves were much less noisy than the 1px, but I didn't hear too much of a difference between 2 and 3px. However, when I tested again with the higher frequencies (2000 cycles/rev) I could hear much more noise on the 3px groove than the 2px.

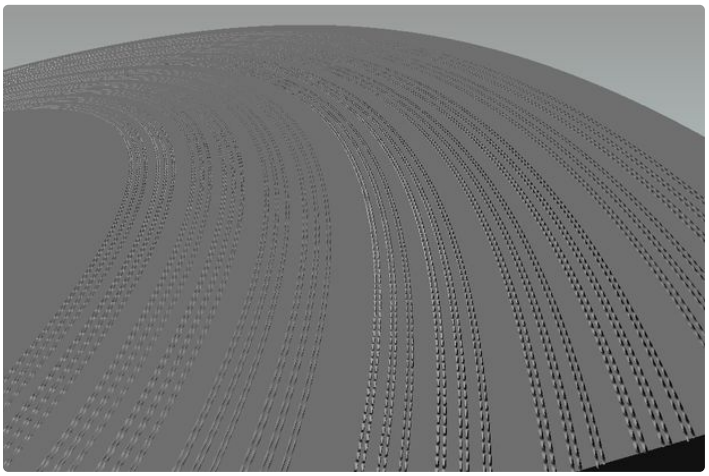
Frequency Range - at 22000 points per revolution, I easily achieved the upper limit of the human vocal range (about 1.1kHz). Theoretically I should be able to reproduce frequencies equal to half my sampling rate. With a sampling rate of 12kHz (calculated in the last step), the highest frequency I can theoretically achieve is 6kHz. I suspect that the movement of the liquid resin during the curing process will prevent me from actually achieving these frequencies, but if I can just get into the 2kHz range it will still sound reasonably good. Based on the tests I've run so far, I think this is possible.

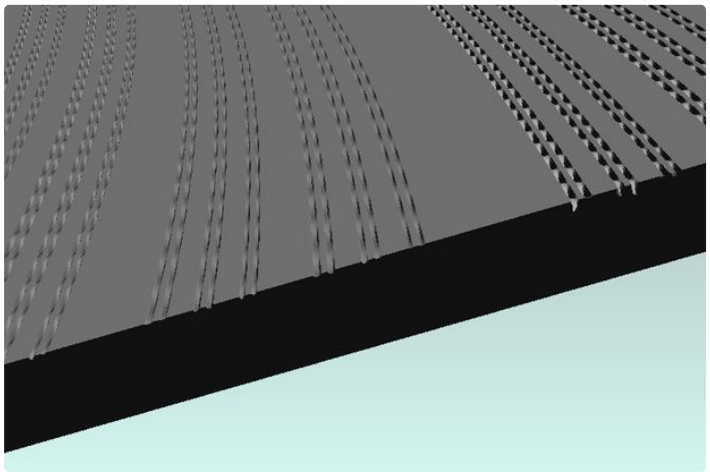
Dimensions - Although it seems like a 12" record should measure 12" in diameter, I found that printing at 12" made the record slightly too large for my record player. I decreased the diameter down to 11.8" and it worked great.

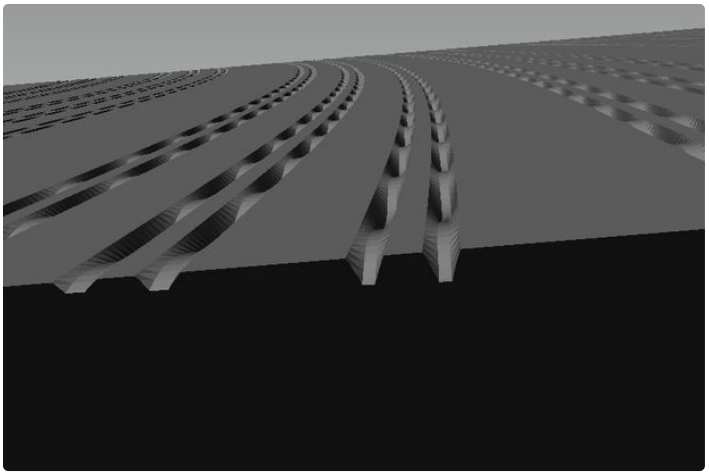
Max file size of ~300MB - Although Processing is capable of producing much larger files, the Objet Software that runs the printers seems to only handle about 300MB of data at a time. It's possible that increased RAM might bring this up to 500mb, but this still does not give me a lot of room to work with. Although this is plenty for normal CAD purposes, I found out that I would have to be very efficient with the way I packed data onto the STL for the final version of my Processing sketch. One problem with my current sketch is that is has a constant angular sampling rate, this means that the same amount of data is used to describe a groove on the outer edge of the record and a groove near the center of the record. Since the groove at the center of the record is much smaller it would a higher resolution than the outer groove, unfortunately, this extra precision goes to waste because the printer has constant DPI across the entire surface of the record. Eventually, I hope to decrease the angular sampling rate of the inner grooves to save storage space and pack as much audio into the STL file as possible.

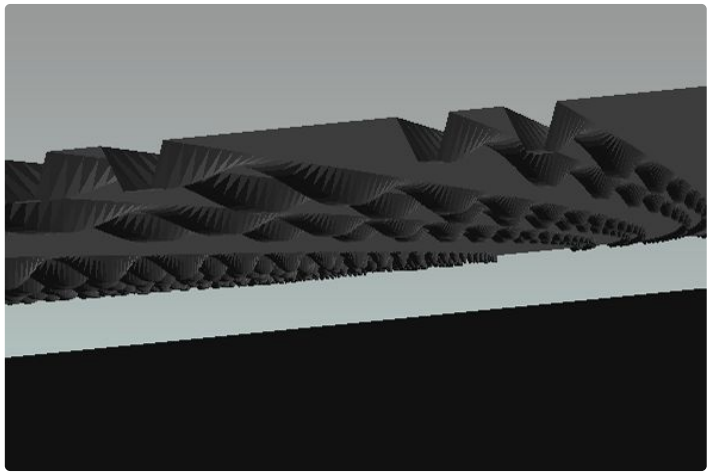










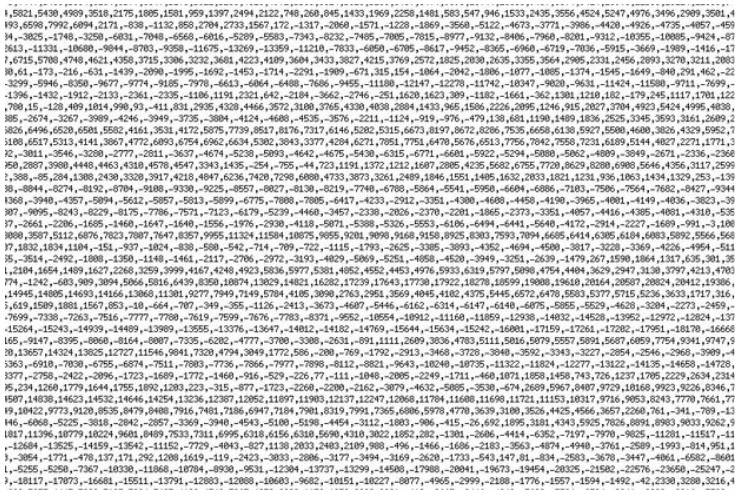


Step 4: Extracting Audio Data With Python

Processing has a library for dealing with audio called Minim, it is included with the more recent versions of Processing IDE. Unfortunately, this library is set up for real time audio applications and does not appear give you an easy way of extracting all the data from an audio file at once (it makes you load it in small buffers piece by piece). Since I could not find an easy way to load my wav files into Processing directly (although I'm sure this must be possible), I've been importing stereo audio in the wav format into Python using the wav library, adding the left and right channels together, centering the data around zero, and exporting the resulting array of int's (separated by commas) to a txt file. Here is my wav to txt Python script (I'm running this in Python 2.5.4):

```
<pre>#Wav to Txt
#by Amanda Ghassaei
#https://www.instructables.com/member/amandaghassaei/

## * This program is free software; you can redistribute it and/or modify
## * it under the terms of the GNU General Public License as published by
## * the Free Software Foundation; either version 3 of the License, or
## * (at your option) any later version.

#this code unpacks and repacks data from:
#16 bit stereo wav file at 44100hz sampling rate
#and saves it as a txt file

import wave
import math
import struct

bitDepth = 8#target bitDepth
frate = 44100#target frame rate

fileName = "audio.wav"#file to be imported (change this)

#read file and get data
w = wave.open(fileName, 'r')
numframes = w.getnframes()

frame = w.readframes(numframes)#w.getnframes()

frameInt = map(ord, list(frame))#turn into array

#separate left and right channels and merge bytes
frameOneChannel = [0]*numframes#initialize list of one channel of wave
for i in range(numframes):
    frameOneChannel[i] = frameInt[4*i+1]*2**8+frameInt[4*i]#separate channels and store one channel in new list
    if frameOneChannel[i] > 2**15:
        frameOneChannel[i] = (frameOneChannel[i]-2**16)
    elif frameOneChannel[i] == 2**15:
        frameOneChannel[i] = 0
    else:
        frameOneChannel[i] = frameOneChannel[i]

#convert to string
audioStr = ''
for i in range(numframes):
    audioStr += str(frameOneChannel[i])
    audioStr += ","#separate elements with comma

fileName = fileName[:-3]#remove .wav extension
text_file = open(fileName+"txt", "w")
text_file.write("%s"%audioStr)
text_file.close()
```

Once I create a text file, I can import the data into Processing and convert it into an STL. I would like to streamline my code so that audio files (wav/mp3) can be loaded directly into Processing, but I have not found a solution yet, any suggestions would be appreciated!



Step 5: Initial Audio Tests (featuring the Pixies)

Finally, it was time to start printing real audio! All my initial audio tests were done with the first 30 seconds from the opening track of one of the greatest albums ever written, The Pixies "Debaser."

In the first test I used the same parameters that I'd had luck with in the sine wave tests:

amplitude: 16 x 16 micron z axis steps
groove width: 2px
groove depth: 3 x 16 micron z axis steps
sampling rate: 11,025hz (one quarter of normal mp3 sampling rate)
groove spacing: 20 pixels (at 600 dpi)

My processing code is below. The code is heavily commented, but here's the overall gist:
An audio file is basically a list of numbers that plot a waveform over time. The data that I pulled from Python in the last step is just that, the list of data points in the audio file. Essentially what I did in this Processing sketch was use this data to set the depth of a long, spiral groove on the record's surface. Later when the needle passes over this groove, its tip will follow this path and actually trace out the original waveform stored in the audio data.

```
<pre>//txt to stl conversion - 3d printable record
//by Amanda Ghassaei
//Dec 2012
//https://www.instructables.com/id/3D-Printed-Record/

/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3 of the License, or
 * (at your option) any later version.
*/


import processing.opengl.*;
import unlekker.util.*;
import unlekker.modelbuilder.*;
import ec.util.*;

String filename = "debaser.txt";

UVertexList recordPerimeterUpper,recordPerimeterLower,recordHoleUpper,recordHoleLower;//storage for perimeter and center hole of record
UVertexList lastEdge;//storage for conecting one groove to the next
UGeometry geo;//storage for stl geometry
```

```
//parameters
float samplingRate = 44100;//(44.1hz audio)
float rpm = 33.3;//rev per min
float secPerMin = 60;//seconds per minute
float rateDivisor = 4;//how much we are downsampling by
float theta;//angle variable
float thetaIter = (samplingRate*secPerMin)/(rateDivisor*rpm);//how many values of theta per cycle
float radius;//variable to calculate radius of grooves
float diameter = 11.8;//diameter of record in inches
float innerHole = 0.286;//diameter of center hole in inches
float innerRad = 2.35;//radius of innermost groove in inches
float outerRad = 5.75;//radius of outermost groove in inches
float grooveSpacing = 20;//pixel spacing of grooves

//record parameters
float recordHeight = 0.04;//height of record in inches
int recordBottom = 0;//height of bottom of record

//variable parameters
float amplitude = 24;//amplitude of signal (in 16 micron steps)
float bevel = 0.5;//bevelled groove edge
float grooveWidth = 3;//in 600dpi pixels
float depth = 6;//measured in 16 microns steps, depth of tops of wave in groove from uppermost surface of record

float incrNum = TWO_PI/thetaIter;//calculcate angular incrementation amount

int grooveNum = 0;//variable for keeping track of how long this will take
int totalSampleNum;

void setup(){

  geo = new UGeometry();//place to store geometery of verticies

  setUpVariables();//convert units, initialize etc
  setUpRecordShape();//draw basic shape of record
  drawGrooves(processAudioData());//draw in grooves

  geo.writeSTL(this, filename + ".stl");//write stl file from geomtery

  exit();
}

float[] processAudioData(){

  //get data out of txt file
  String rawData[] = loadStrings(filename);
  String rawDataString = rawData[0];
  float audioData[] = float(split(rawDataString,','));//separated by commas

  //normalize audio data to given bitdepth
  //first find max val
  float maxval = 0;
  for(int i=0;i<audioData.length;i++){
    if (abs(audioData[i])>maxval){
      maxval = abs(audioData[i]);
    }
  }
  //normalize amplitude to max val
  for(int i=0;i<audioData.length;i++){
    audioData[i]*=amplitude/maxval;
  }

  return audioData;
}

void setUpVariables(){

  //convert everything to inches
  float micronsPerInch = 25400;//scalingfactor
  float dpi = 600;
  byte micronsPerLayer = 16;//microns per vertical print layer

  grooveSpacing /= dpi;
  amplitude = amplitude*micronsPerLayer/micronsPerInch;
```

```
  depth = depth*micronsPerLayer/micronsPerInch;
  grooveWidth /= dpi;

}

void setUpRecordShape(){

  //set up storage
  recordPerimeterUpper = new UVertexList();
  recordPerimeterLower = new UVertexList();
  recordHoleUpper = new UVertexList();
  recordHoleLower = new UVertexList();

  //get verticies
  for(theta=0;theta<TWO_PI;theta+=incrNum){
    //outer edge of record
    float perimeterX = diameter/2+diameter/2*cos(theta);
    float perimeterY = diameter/2+diameter/2*sin(theta);
    recordPerimeterUpper.add(perimeterX,perimeterY,recordHeight);
    recordPerimeterLower.add(perimeterX,perimeterY,recordBottom);
    //center hole
    float centerHoleX = diameter/2+innerHole/2*cos(theta);
    float centerHoleY = diameter/2+innerHole/2*sin(theta);
    recordHoleUpper.add(centerHoleX,centerHoleY,recordHeight);
    recordHoleLower.add(centerHoleX,centerHoleY,recordBottom);
  }

  //close vertex lists (closed loops)
  recordPerimeterUpper.close();
  recordPerimeterLower.close();
  recordHoleUpper.close();
  recordHoleLower.close();

  //connect verticies
  geo.quadStrip(recordHoleUpper,recordHoleLower);
  geo.quadStrip(recordHoleLower,recordPerimeterLower);
  geo.quadStrip(recordPerimeterLower,recordPerimeterUpper);

  //to start, outer edge of record is the last egde we need to connect to with the outmost groove
  lastEdge = new UVertexList();
  lastEdge.add(recordPerimeterUpper);

  println("record drawn, starting grooves");
  grooveNum = 0;//variable for keeping track of how much longer this will take

}

void drawGrooves(float[] audioData){

  UVertexList grooveOuterUpper,grooveOuterLower,grooveInnerUpper,grooveInnerLower;//groove verticies
  UVertexList stop1,stop2;//storage for very beginning and end of sprial groove

  //set up storage
  grooveOuterUpper = new UVertexList();
  grooveOuterLower = new UVertexList();
  grooveInnerUpper = new UVertexList();
  grooveInnerLower = new UVertexList();
  stop1 = new UVertexList();
  stop2 = new UVertexList();

  //DRAW GROOVES
  radius = outerRad;//outermost radius (at 5.75") to start
  float radIncr = (grooveSpacing+grooveWidth)/thetaIter;//calculate radial incrementtion amount
  int samplenum = 0;
  int totalgroovenum = int(audioData.length/(rateDivisor*thetaIter));

  //first draw starting cap
  theta = 0;
  float sineTheta = sin(theta);
  float cosineTheta = cos(theta);
  //calculate height of groove
  float grooveHeight = recordHeight-depth-amplitude+audioData[int(rateDivisor*samplenum)];
  stop1.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);//outerupper
  stop2.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);//outerlower
  stop2.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);//innerlower
```

```
  stop1.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);//innerupper
  //draw triangles
  geo.quadStrip(stop1,stop2);

  //then spiral groove
  while (radius>innerRad && rateDivisor*samplenum<(audioData.length-rateDivisor*thetaIter+1)){//while we still have audio to write and we have not reached the innermost
groove

    //clear lists
    grooveOuterUpper.reset();
    grooveOuterLower.reset();
    grooveInnerUpper.reset();
    grooveInnerLower.reset();

    for(theta=0;theta<TWO_PI;theta+=incrNum){//for theta between 0 and 2pi

      sineTheta = sin(theta);
      cosineTheta = cos(theta);

      //calculate height of groove
      grooveHeight = recordHeight-depth-amplitude+audioData[int(rateDivisor*samplenum)];
      samplenum++;//increment sample num

      grooveOuterUpper.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);
      grooveOuterLower.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);
      grooveInnerLower.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);
      grooveInnerUpper.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);

      radius -= radIncr;

    }

    //add last value to grooves to complete one full rev
    theta = 0;
    sineTheta = sin(theta);
    cosineTheta = cos(theta);

    //calculate height of groove
    grooveHeight = recordHeight-depth-amplitude+audioData[int(rateDivisor*samplenum)];

    grooveOuterUpper.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);
    grooveOuterLower.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);
    grooveInnerLower.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);
    grooveInnerUpper.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);

    //connect verticies
    geo.quadStrip(lastEdge,grooveOuterUpper);
    geo.quadStrip(grooveOuterUpper,grooveOuterLower);
    geo.quadStrip(grooveOuterLower,grooveInnerLower);
    geo.quadStrip(grooveInnerLower,grooveInnerUpper);

    //set new last edge
    lastEdge.reset();//clear old data
    lastEdge.add(grooveInnerUpper);

    //complete beginning cap if necessary
    if (grooveNum==1){
      //clear stop2
      stop2.reset();
      stop2.add(diameter/2+diameter/2*cosineTheta,diameter/2+diameter/2*sineTheta,recordHeight);//outer perimeter[0]
      stop2.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);//outer groove edge [2pi]
      //draw triangles
      geo.quadStrip(stop1,stop2);
    }

    //tell me how much longer
    grooveNum++;
    print(grooveNum);
    print(" of ");
    print(totalgroovenum);
    println(" grooves drawn");
  }

  //draw end cap of spiral groove
  stop1.reset();
  stop2.reset();
```

```
  stop2.reset();
  stop1.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);//outeruppter
  stop2.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);//outerlower
  stop2.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);//innerlower
  stop1.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);//innerupper
  //draw triangles
  geo.quadStrip(stop1,stop2);
  stop2.reset();
  stop2.add(grooveInnerUpper.first());//innerupper[0]
  stop2.add(diameter/2+innerHole/2*cosineTheta,diameter/2+innerHole/2*sineTheta,recordHeight);//innerhole[0]
  //draw triangles
  geo.quadStrip(stop1,stop2);

  geo.quadStrip(lastEdge,recordHoleUpper);//close remaining space between last groove and center hole

}
```

and here is the result:

https://www.youtube.com/embed/2B9fsypKo7E

Success! You can clearly hear The Pixies coming through, but the signal to noise ratio is not great. In my next test I amplified the original audio signal a little bit before sending it to my Processing sketch. This way some of the louder drum sections would get slightly clipped and allow the overall amplitude of the normalized signal to get a little larger. Here's what that sounds like:

https://www.youtube.com/embed/0VR0d1fAB7E

Signal to noise is getting better, I added a little more audio to this file so that you can start to hear Frank Black's vocals coming in. Next I increased the amplitude of the signal to see if I could get a better signal out. In my sine tests I thought that an amplitude of 16 was plenty loud, but not so large that it caused excessive distortion in the signal. Since the Pixies signal is not always spanning the full amplitude allowed by my program I increased the amplitude of the algorithm (most of the time the waveform is hovering around half of its peak amplitude, only the drums are able to kick the signal up to full amplitude). This may cause some extra distortion on the drum beats, but since drums are already pretty noisy I was ok with that. Here's the result of amplitude 32:

https://www.youtube.com/embed/Nb2DXALE2YQ

Signal to noise is better, but there is quite a bit of distortion. I decreased the amplitude to 24 next:

https://www.youtube.com/embed/MhArntzIzVs

This sounds a lot better. Good signal to noise without too much distortion. Next I made a slight edit to my code to minimize the amount of data packed into the stl file. In the previous examples I created some space between grooves, basically a flat surface parallel to the top of the record. In the code below I removed this space and used the last upper edge of the previous groove as the upper edge of the next groove. The difference in the model is shown in fig 2.

```
<pre>//txt to stl conversion - 3d printable record
//by Amanda Ghassaei
//Dec 2012
//https://www.instructables.com/id/3D-Printed-Record/

/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3 of the License, or
 * (at your option) any later version.
*/


import processing.opengl.*;
import unlekker.util.*;
import unlekker.modelbuilder.*;
import ec.util.*;

String filename = "bluemonday.txt";

UVertexList recordPerimeterUpper,recordPerimeterLower,recordHoleUpper,recordHoleLower;//storage for perimeter and center hole of record
UVertexList lastEdge;//storage for conecting one groove to the next
UGeometry geo;//storage for stl geometry

//parameters
float samplingRate = 44100;//(44.1khz audio)
float rpm = 33.3;//rev per min
float secPerMin = 60;//seconds per minute
float rateDivisor = 4;//how much we are downsampling by
float theta;//angle variable
float thetaIter = (samplingRate*secPerMin)/(rateDivisor*rpm);//how many values of theta per cycle
float radius;//variable to calculate radius of grooves
float diameter = 11.8;//diameter of record in inches
float innerHole = 0.286;//diameter of center hole in inches
float innerRad = 2.35;//radius of innermost groove in inches
float outerRad = 5.75;//radius of outermost groove in inches

//record parameters
float recordHeight = 0.04;//height of record in inches
int recordBottom = 0;//height of bottom of record

//variable parameters
float amplitude = 24;//amplitude of signal (in 16 micron steps)
float bevel = 0.5;//bevelled groove edge
float grooveWidth = 2;//in 600dpi pixels
```

```
float depth = 6;//measured in 16 microns steps, depth of tops of wave in groove from uppermost surface of record

float incrNum = TWO_PI/thetaIter;//calculcate angular incrementation amount

int grooveNum = 0;//variable for keeping track of how long this will take
int totalSampleNum;

void setup(){

  geo = new UGeometry();//place to store geometery of verticies

  setUpVariables();//convert units, initialize etc
  setUpRecordShape();//draw basic shape of record
  drawGrooves(processAudioData());//draw in grooves

  geo.writeSTL(this, filename + ".stl");//write stl file from geomtery

  exit();
}

float[] processAudioData(){

  //get data out of txt file
  String rawData[] = loadStrings(filename);
  String rawDataString = rawData[0];
  float audioData[] = float(split(rawDataString,','));//separated by commas

  //normalize audio data to given bitdepth
  //first find max val
  float maxval = 0;
  for(int i=0;i<audioData.length;i++){
    if (abs(audioData[i])>maxval){
      maxval = abs(audioData[i]);
    }
  }
  //normalize amplitude to max val
  for(int i=0;i<audioData.length;i++){
    audioData[i]*=amplitude/maxval;
  }

  return audioData;
}

void setUpVariables(){

  //convert everything to inches
  float micronsPerInch = 25400;//scalingfactor
  float dpi = 600;
  byte micronsPerLayer = 16;//microns per vertical print layer

  amplitude = amplitude*micronsPerLayer/micronsPerInch;
  depth = depth*micronsPerLayer/micronsPerInch;
  grooveWidth /= dpi;

}

void setUpRecordShape(){

  //set up storage
  recordPerimeterUpper = new UVertexList();
  recordPerimeterLower = new UVertexList();
  recordHoleUpper = new UVertexList();
  recordHoleLower = new UVertexList();

  //get verticies
  for(theta=0;theta<TWO_PI;theta+=incrNum){
    //outer edge of record
    float perimeterX = diameter/2+diameter/2*cos(theta);
    float perimeterY = diameter/2+diameter/2*sin(theta);
    recordPerimeterUpper.add(perimeterX,perimeterY,recordHeight);
    recordPerimeterLower.add(perimeterX,perimeterY,recordBottom);
    //center hole
    float centerHoleX = diameter/2+innerHole/2*cos(theta);
    float centerHoleY = diameter/2+innerHole/2*sin(theta);
    recordHoleUpper.add(centerHoleX,centerHoleY,recordHeight);
    recordHoleLower.add(centerHoleX,centerHoleY,recordBottom);
```

```
    recordHoleLower.add(centerHoleX,centerHoleY,recordBottom);
  }

  //close vertex lists (closed loops)
  recordPerimeterUpper.close();
  recordPerimeterLower.close();
  recordHoleUpper.close();
  recordHoleLower.close();

  //connect verticies
  geo.quadStrip(recordHoleUpper,recordHoleLower);
  geo.quadStrip(recordHoleLower,recordPerimeterLower);
  geo.quadStrip(recordPerimeterLower,recordPerimeterUpper);

  //to start, outer edge of record is the last egde we need to connect to with the outmost groove
  lastEdge = new UVertexList();
  lastEdge.add(recordPerimeterUpper);

  println("record drawn, starting grooves");
  grooveNum = 0;//variable for keeping track of how much longer this will take

}

void drawGrooves(float[] audioData){

  UVertexList grooveOuterUpper,grooveOuterLower,grooveInnerUpper,grooveInnerLower;//groove verticies
  UVertexList stop1,stop2;//storage for very beginning and end of sprial groove

  //set up storage
  grooveOuterUpper = new UVertexList();
  grooveOuterLower = new UVertexList();
  grooveInnerUpper = new UVertexList();
  grooveInnerLower = new UVertexList();
  stop1 = new UVertexList();
  stop2 = new UVertexList();

  //DRAW GROOVES
  radius = outerRad;//outermost radius (at 5.75") to start
  float radIncr = (grooveWidth+2*bevel*amplitude)/thetaIter;//calculate radial incrementation amount
  int samplenum = 0;
  int totalgroovenum = int(audioData.length/(rateDivisor*thetaIter));

  //first draw starting cap
  theta = 0;
  float sineTheta = sin(theta);
  float cosineTheta = cos(theta);
  //calculate height of groove
  float grooveHeight = recordHeight-depth-amplitude+audioData[int(rateDivisor*samplenum)];
  stop1.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);//outerupper
  stop2.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);//outerlower
  stop2.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);//innerlower
  stop1.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);//innerupper
  //draw triangles
  geo.quadStrip(stop1,stop2);

  //then spiral groove
  while (radius>innerRad && rateDivisor*samplenum<(audioData.length-rateDivisor*thetaIter+1)){//while we still have audio to write and we have not reached the innermost
groove

    //clear lists
    grooveOuterUpper.reset();
    grooveOuterLower.reset();
    grooveInnerUpper.reset();
    grooveInnerLower.reset();

    for(theta=0;theta<TWO_PI;theta+=incrNum){//for theta between 0 and 2pi

      sineTheta = sin(theta);
      cosineTheta = cos(theta);

      //calculate height of groove
      grooveHeight = recordHeight-depth-amplitude+audioData[int(rateDivisor*samplenum)];
      samplenum++;//increment sample num

      if (grooveNum==0){
        grooveOuterUpper.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);
```

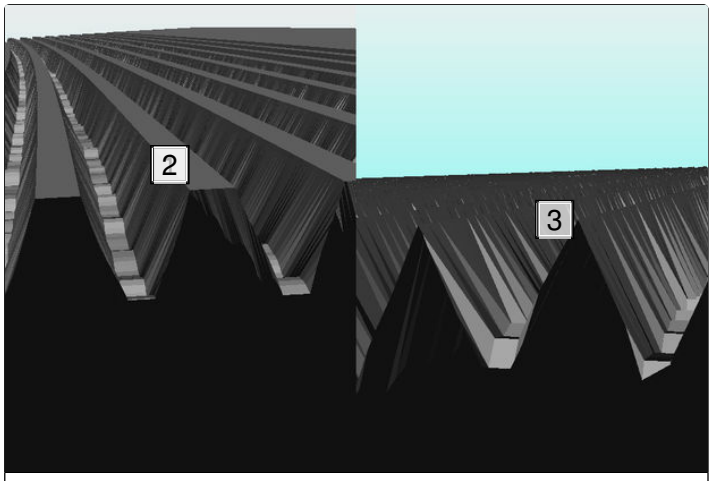
```
    }
    grooveOuterLower.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);
    grooveInnerLower.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);
    grooveInnerUpper.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);

    radius -= radIncr;

  }

  //add last value to grooves to complete one full rev
  theta = 0;
  sineTheta = sin(theta);
  cosineTheta = cos(theta);

  //calculate height of groove
  grooveHeight = recordHeight-depth-amplitude+audioData[int(rateDivisor*samplenum)];

  if (grooveNum==0){
    grooveOuterUpper.add(grooveInnerUpper.first());//grooveOuterUpper.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sine
Theta),recordHeight);
  }
  grooveOuterLower.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);
  grooveInnerLower.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);
  grooveInnerUpper.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);

  //connect verticies
  if (grooveNum==0){//if joining a roove to the edge of the record
    geo.quadStrip(lastEdge,grooveOuterUpper);
    geo.quadStrip(grooveOuterUpper,grooveOuterLower);
  }
  else{//if joining a groove to another groove
    geo.quadStrip(lastEdge,grooveOuterLower);
  }
  geo.quadStrip(grooveOuterLower,grooveInnerLower);
  geo.quadStrip(grooveInnerLower,grooveInnerUpper);

  //set new last edge
  lastEdge.reset();//clear old data
  lastEdge.add(grooveInnerUpper);

  //complete beginning cap if necessary
  if (grooveNum==0){
    //clear stop2
    stop2.reset();
    stop2.add(diameter/2+diameter/2*cosineTheta,diameter/2+diameter/2*sineTheta,recordHeight);//outer perimeter[0]
    stop2.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);//outer groove edge [2pi]
    //draw triangles
    geo.quadStrip(stop1,stop2);
  }

  //tell me how much longer
  grooveNum++;
  print(grooveNum);
  print(" of ");
  print(totalgroovenum);
  println(" grooves drawn");
 }

 //draw end cap of spiral groove
 stop1.reset();
 stop2.reset();
 stop1.add((diameter/2+(radius+amplitude*bevel)*cosineTheta),(diameter/2+(radius+amplitude*bevel)*sineTheta),recordHeight);//outeruppter
 stop2.add((diameter/2+radius*cosineTheta),(diameter/2+radius*sineTheta),grooveHeight);//outerlower
 stop2.add((diameter/2+(radius-grooveWidth)*cosineTheta),(diameter/2+(radius-grooveWidth)*sineTheta),grooveHeight);//innerlower
 stop1.add((diameter/2+(radius-grooveWidth-amplitude*bevel)*cosineTheta),(diameter/2+(radius-grooveWidth-amplitude*bevel)*sineTheta),recordHeight);//innerupper
 //draw triangles
 geo.quadStrip(stop1,stop2);
 stop2.reset();
 stop2.add(lastEdge.last());//innerupper[0]
 stop2.add(diameter/2+innerHole/2*cosineTheta,diameter/2+innerHole/2*sineTheta,recordHeight);//innerhole[0]
 //draw triangles
 geo.quadStrip(stop1,stop2);

 geo.quadStrip(lastEdge,recordHoleUpper);//close remaining space between last groove and center hole
```

```
}
```

And here's what it sounds like:

```
https://www.youtube.com/embed/rhlf75dHjJo
```

I was really happy with the way this came out, this is the code I used moving forward.



1. slight spacing between grooves
2. slight spacing between grooves
3. no space between grooves (more efficiently packed data)

Step 6: Records

After nearly a week of printing, frantically running around downtown SF, and generally fighting with technology, I've got some reasonably good sounding audio to share with you:

//www.youtube.com/embed/PzI5MJLj7KI

//www.youtube.com/embed/BKRI57G2HiA

//www.youtube.com/embed/3AIxhxa9r28

//www.youtube.com/embed/WCAI40HRfQY

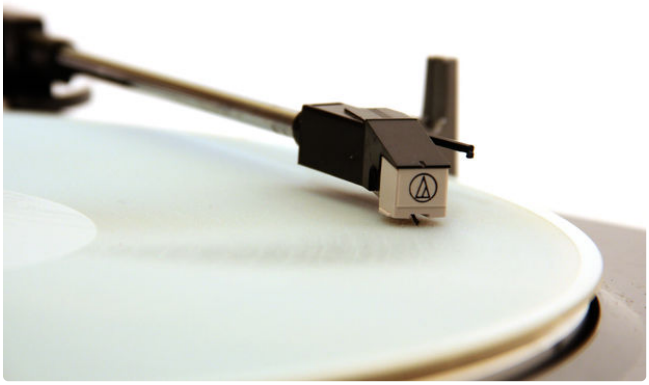
//www.youtube.com/embed/DkRo6im_5ig

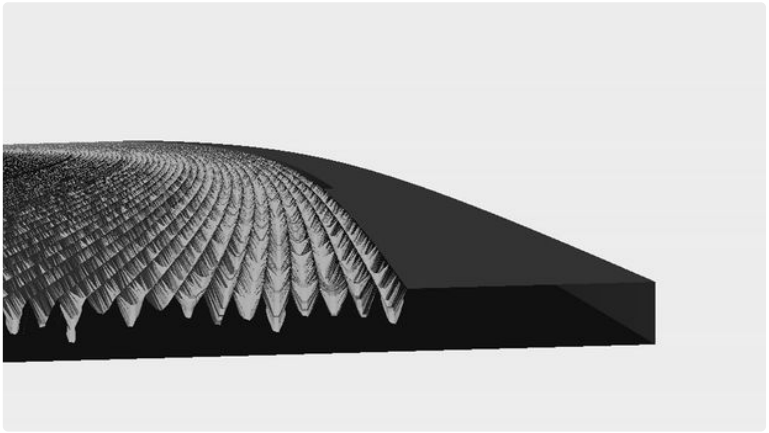
You'll notice that all of these prints are only about a minute long- this was due to some issues I was having with file size and RAM. I'm currently working on troubleshooting these problems, but at the moment the largest file I can print out is about 250MB, or a little over a minute of audio. Each size has the potential to fit about 6 minutes of audio, this amounts to a file containing about 1.5GB of data.

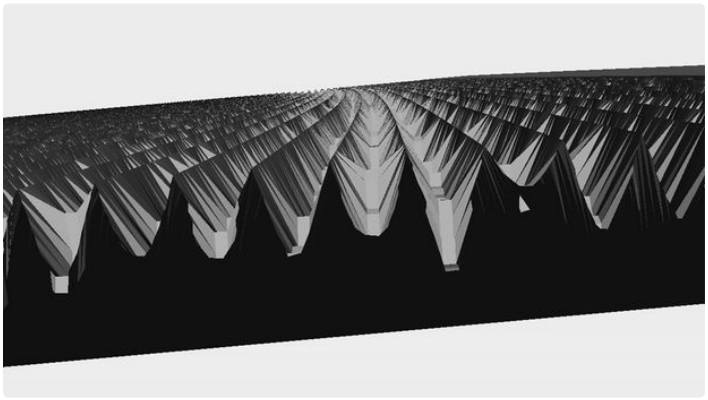
Some of these files are up on The Pirate Bay and the 123D gallery for anyone to download. (Pirate Bay recently introduced new section on their site for sharing 3d designs called Physibles, many of the models are ready for 3D printing or other forms of digital fabrication). Unfortunately, some of my files were so large that I had to down-sample even farther, to 9kHz, to get them to export before my computer crashed. I'm still not done generating all the files I would like to make, it takes a bit of time to process and handle such large amounts of data, so things are moving along relatively slowly. Check back for updates, and if you download any of these, please seed them!

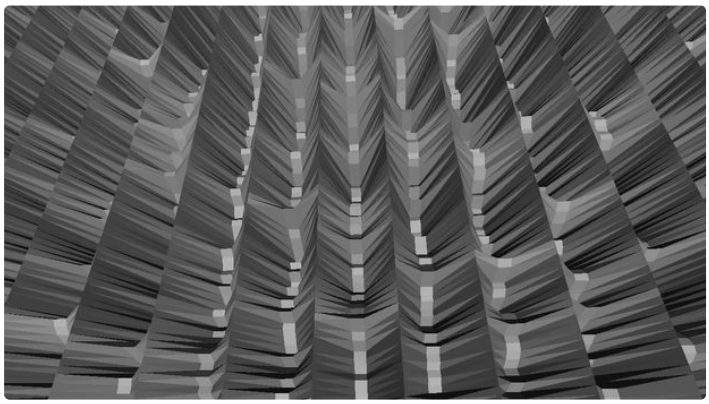
Pink Floyd - Dark Side of the Moon (4 discs, 7 sides) Supposedly, there is a factory in germany that does nothing but press copies of dark side of the moon, maybe one day 3d printers will put that factory out of business, but for now I've had to split the album up onto seven sides of four discs to even get it to fit...
New Order - Blue Monday Single(1 disc, 2 sides)The top selling 12" single of all time. Side a is Blue Monday in its nearly 8 minute entirety, I had to cheat a little and extend the grooves into the space where the label should go to pull this off, but it's not too bad, and side b is a remix of Blue Monday called The Beach, just like the original release.
White Stripes - Fell in Love with a Girl(1 disc, 1 side) This is song is not even 2 min long, so I exported it at slightly higher res- 45rpm with a 16kHz sampling rate. Now that our 3D printing facility has an upgraded computer, I'd like to print out a bunch of short songs like this at 45rpm.

I made all these files at half the thickness of a regular record. That way, if you ever find a way to print them, you can glue two of them together to make a double sided record.









Step 7: Make Your Own Records

You can convert your own audio files into 3D STL models in ten easy steps:

1. Download Processing.

2. Download the ModelBuilder library for Processing. I used version 0007a03.

3. Unzip the Modelbuilder library .zip and copy the folder inside called "modelbuilder". Unzip the processing .zip and go to Processing>modes>java>libraries and paste the "modelbuilder" folder in the "libraries" folder.

4. Download Python 2.5.4.

5. Download Audacity.

6. Download the code from GitHub (you can download the files as a zip by clicking on the cloud button). Open the folder called Processing3DPrintedRecord.

7. Open an audio file of your choice with Audacity. Use Effect>Amplify to amplify the signal as much as you can without noticeable effects of clipping (you will be able to get away with some clipping, and remember this is not crystal clear audio anyway). Make sure there is 2 sec of blank audio at the end of the track so that nothing gets clipped. Keep the audio under 6 min. File>Export this file and save it in the "Processing3DPrintedRecord" folder as a wav file.

8. Open the Python file called "wavtotxt". Copy the file name of the file you just saved in the line:

fileName = "your_file_name_here.wav"

Hit Run>RunModule, after a minute or two you will have a .txt file saved in the Processing3DPrintedRecord folder.

9. Open the Processing Sketch. In File>Preferences check the box that says "increase the maximum available memory to" and write in the amount of available ram on your computer. I've found that my laptop with 4GB RAM can handle audio files up to 1.5 min long. For longer files you will need to use a computer with 12-16GB of RAM. If Processing starts and then gets stuck or crashes, it is a RAM issue and you will need to move to a machine with more RAM.

10. Change the name of the import file in the Processing sketch to your txt file name:

String filename = "your_file_name_here.txt";

Close all other programs running on your computer and run the Processing sketch Sketch>Run. After a few seconds you will see "record drawn, starting grooves" appear at the bottom of the Processing window. After some more time you will get updates on the status of the sketch: "3 of 85 grooves drawn". Eventually Processing will tell you that it is writing your STL file and when it is done it will say "Closing 'name_of_your_file.stl' and tell you haw many faces are in the file. You can find the finished file in the Processing3DPrintedRecord folder.

Once you've made them, post them! I've been posting mine on the 123D gallery and the Pirate Bay. Enjoy, and let me know if you have questions or need help getting this to work. I've tested this process in both Windows (64 bit) and Mac OS, though I'd imagine it will work for Linux as well.

https://www.instructabl...

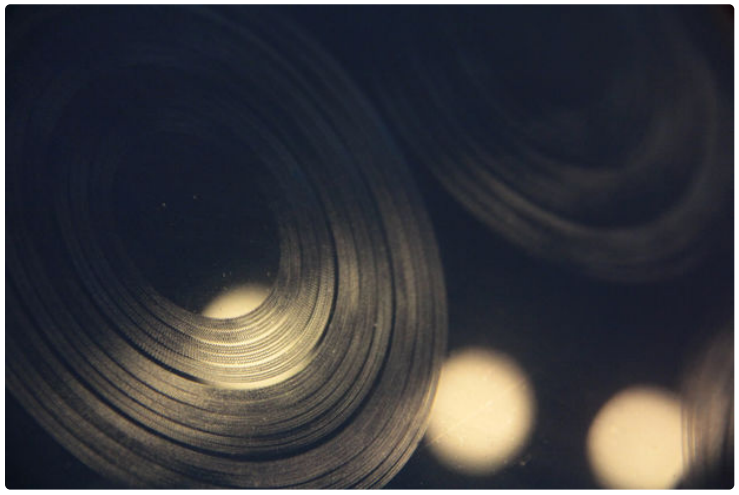
Download

Step 8: Future Work

We're currently trying to upgrade our computer setup so that we will be able to print out files larger than 250MB. Eventually I'd like to actually print physical copies of some of the files that I posted in the last step. I'm also interested in hearing how the resolution difference between the outer and inner grooves of the record (explained in step 2) manifest themselves in the audio output.

Soon I'd also like to experiment with some more creative applications of this technology. For example, printing out a record with many adjacent closed loops of ~2second looping samples. This way you could set your needle down in one groove and listen to a loop repeat over and over, then tap the needle to the side to switch to another loop. Assuming all the the loops have a similar time signature, you could turn this record into a cool, interactive sample mixer.

I'm currently working on another project that takes audio data and outputs a vector cutting path in the shape of a record (pictured above). I'm planning on cutting this record with a laser cutter on acrylic. Unfortunately, we've been having some trouble with our lasers recently, but I hope to get the project up in the first few weeks of January, as I have most of the code done. I'm excited about this project because it has the potential to be a lot more useful to ordinary(ish) people. The vector files are much easier and faster to generate, and the whole process uses cheaper materials and tools that a decent amount of people have access to these days. I still haven't done enough testing to say how it will compare to my 3D printed record, but I'm fairly confident it will work.



I received an email today from someone asking me about how far I managed to go with replicating Amanda's process, so I thought I'd share my experience in more details.

Note that this experience below occurred in December 2014.

My goal was to try print a 3mn song on 1 side of a 12 inches vinyl and see what is the best quality and output I could achieve, noticing that it has been already more than 2 years since Amanda did her project, so I assumed that the technology would have progressed rapidly.

Amanda's instructions and provided programs help you generate the binary file, then create a 3D model (STL file) for a one sided vinyl with that song on it (eq'd specifically for vinyl previously as per the video). At this stage, I don't know if you can even generate a two sided vinyl STL model but this didn't matter to me for that experience.

I downsampled the file to 12hz as per her initial instructions and default settings. The outcome was a file that was 700MB. Generating this file alone requires HUGE amount of processing power. Note that I'm using a Mac Book Pro with 16GB of ram, solid state drive and a very powerful processor. It was nevertheless a painful process for the computer to generate the file. See attached an image of the model. You'll note that the entire surface is not covered, which I couldn't quite figure out why. I assumed that if it did take the entire width, the grooves would be a be bigger and therefore more tolerant to the level of precision currently available via 3D printing.

I then tried changing the downsampling rate to something higher (I started at 24hz), so I could get to a quality closer to what I'm after (i.e. that can be released). It looked something like this:

Unfortunately, the file couldn't even get generated. My computer would hang permanently. That wasn't even trying 44hz which is what I was ideally after...

To note: the output STL file had dimensions for X and Y initially rendering in mm instead of inches for some reason so something to keep an eye out on if you try that technique using Amanda's program (unless it had to do with the program I used to render the shape - I then had to update the dimensions manually to the equivalent in mm).

I stuck to my 12hz downsampling rate then and contacted the most advanced 3D printing company in Melbourne I could find (3Dsystems.com). They had by very far the best machines I could find on the market for once off printing. I figured this would provide me already with a significant improvement of quality from what Amanda had produced.

The machine used was SLA Flex Printer with XHD 0.050 mm layers. Resolutions were uniform on all 3 axis, XYZ with all parts having an accuracy of 0.001 – 0.002 inch per inch of part dimension. This was by far the best printer I could find. The material used was 'Accura Xtreme'.

Unfortunately, the machine and vinyl would break before it could be fully printed, the file being far too large. Files are generraly a few dozens MB at most, and Amanda herself mentions that 300MB is already pushing it.

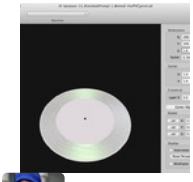
Doing some further research, I then thought that the model may be using a lot of polygons which are not required to print all the details. I therefore asked Amanda if she thought we could use a technique called decimation to remove useless polygons and therefore make the file lighter, but she confirmed that the program she created was pretty efficient. Considering how precise vinyls are in terms of geometry, this makes perfect sense.

Note that the price quoted to me was about US$250 for 1 copy (price goes down obviously as you order more). Copy of the quote attached.
As of today, there therefore doesn't seem to be a way to print even one single song in a decent quality. The song I used was just below 3mn and in mp3 format.

I'm not sure whether there are elements I could have tweaked to change the output (type of printer or material used, settings in the file), but it seemed that as of early 2015, we were still a long way to being able to 3D print vinyls from the comfort of our home, or as an alternative manufacturing process to the vinyl plants (which was the main goal of my investigation here).

If anyone has tried similar experiences and maybe had better outcomes, I'd love to hear about it.

Cheers, Charly



cool.. but $250 seems too high, any option?

- That's not really for me to answer that. You might find 3D printing companies that can do it for less with same or better printer specs, I haven't explored this any further since I posted this comment 2 years ago. That was just for 1 copy though, additional ones were definitely cheaper to make.

thanks

Phew! Don't give up! I managed to get it to work after some trial and error and re-reading the instructions. Most of my problems came about from using different versions of Windows. I found I could only get it to work on my desktop, not my laptop.

thanks for posting this, lots of good info here.

- Hello Amanda!

I represent a national touring band that wants to talk with you about your 3D printed vinyl concept. They want to release plans for people to print a new song for record store day. I'd love to have a quick chat about how feasible this may be and if you'd ever be interested in working with us? Or perhaps recommending someone to work with as I'm sure you're very busy...All the best to you!

Vance Anderson
vanderson7@me.com

- hi amanda,

great job with this (and the laser cut record instructable). i have a few questions about customizing your code for a less precise 3d printer than your objet.

i've gathered i should be changing this code:

```
//convert everything to inches
float micronsPerInch = 25400;//scalingfactor
float dpi = 600;//objet printer prints at 600 dpi
byte micronsPerLayer = 16;//microns per vertical print layer
```

and replace the dpi and layer thickness with specs from the target printer.

do you (or does anyone else reading this) have any experience printing this with with a stratasys uprint plus? it has a layer thickness of 254 microns, so i'm not sure how drastically i might break your code if i increase the micronsPerLayer variable exponentially. are there any other portions of the code i should be careful of in this case?

also, this might be more of an open question to anyone else reading this, but how would i convert the x y resolution of the uprint plus, which gives a minimum wall/feature thickness of .036"/914 microns (and a supposed .020"/508 microns bead diameter) into a DPI value?

thanks again for these two great projects!

- jasper

Hi Jasper - did you try to pritn it up according the amanda´s advice? Did it work?
Thanks Laddy

Im working on something similar - and as a model it looks good - but i´m waiting for a new printer...

cool! I'd recommend changing to these variables:

float amplitude = 24;//you'll want something between 2-5
float bevel = 1;//bevelled groove edge
float grooveWidth = 2;
float depth = 2;/depth of tops of wave in groove from uppermost surface of record

float recordHeight = 0.1;//height of record in inches

//convert everything to inches
float micronsPerInch = 25400;//scalingfactor
float dpi = 50;// 1.0/0.02 = 50
byte micronsPerLayer = 254;//microns per vertical print layer

I'd also recommend changing the rpm to 45 or 78 to get the most out of the lowered x/y res you are printing with.

I'm not sure it will work, but it's definitely worth a try! be sure to pisk something very bass-heavy to start, lower frequencies hold up better to this process than higher ones.

Hi! I want to make this fantastic idea in a CNC machine. I already have a CNC machine, but I do not know no code/program to upload

Congratulations. A fantastic tutorial, very interesting. It is also very complete, quality content with illustrative images.

It is to admire the people who share their time and knowledge altruistically with others. Congratulations and continue like this.

Greetings and many thanks for sharing

Hi Amanda,
This is a fantastic project - Thanks for sharing.
I got this error message:

The function close() does not exist

at line 124: recordPerimeterUpper.close();

Any idea on how to fix this.

- Here's an interesting question to put to you and the community. How could the Processing Code be modified to generate the groove on a cylinder, instead of a flat record. I"m doing a bit of material research for a class and want to try 3D printing working phonograph cylinders. Your record printing seems to be in line with what I"m trying to do, but I'm not sure how to go about generating the file. Any suggestions or insite would be appreciated.

In addition, this 3D printer has a dual extrusion having a closed environment. It includes a sliding door that simplyprovides you withaccess to your prints.

Allthat3D.com/Cheap-3D-Printer/

- Hello Amanda,

The vinyl 3D generate with record generator as somme error.

when i repair the STL with Slic3r there are a probleme of layer. How resolve this probleme and use the STL to gcode.

Do you have this probleme to print the STL?

- Hello Amanda,

I resolve my probleme and i made it.

Could you tel me de good parameter for a rmp 33.3

I have take the following parametter an i have a little disque. 11.8 m x 11.8 m

```
rpm = 33.3;//rev per min
float secPerMin = 60;//seconds per minute
float rateDivisor = 4;//how much we are downsampling by
float theta;//angle variable
float thetaIter = (samplingRate*secPerMin)/(rateDivisor*rpm);//how many values of theta per cycle
float radius;//variable to calculate radius of grooves
float diameter = 11.8;//diameter of record in inches
float innerHole = 0.286;//diameter of center hole in inches
float innerRad = 2.35;//radius of innermost groove in inches
float outerRad = 5.75
```

many thanks.

- draw record and groove geometry Looks awesome

Thanks in advance

- Good project

I did not find the program or code to do the following (i download the processing 3 softwar):

- draw record and groove geometry

- export model in STL format (the link is break)

could you help me to made your project,

https://github.com/amandaghassaei/3DPrintedRecord

- thank for your repy

- Dear Amanda,
Thanks for this brillant project.

Is there any way to use it as sort to get something more extruded, more like a sculpture?

I'm thinking about these prints made from different songs (http://www.realitat.com/microsonic/portishead.html)

Good post. I learn some thing tougher on distinct blogs everyday. Most commonly it really is stimulating to learn to read content material from other writers and exercise a specific thing there.

http://enablerslove.tumblr.com/

First off, you've got bloody good taste in music. Thank you for not printing any Beatles tracks . . . that would be fine, but . . .

Next, I'm shocked you can hear anything at all printing this stuff. It's fascinating. Obviously the higher the frequency, the higher the resolution needed. Prints of older music containing less fidelity will naturally sound closer to what we're used to hearing, due to the lack of upper harmonics.

At any rate, what I'm really wondering here is, someone needs to write a new slicer, or whatever software, that can route the printer in circles. I clearly hear the "sawing" of the needle traversing the lines that are created by the printhead going at 90º angles all the time. Shouldn't we be trying to print this stuff in a spiral? Just a thought. The resolution requirements would actually be less if the material were deposited in a spiral. The resolution requirements of printing curves using only right angles is far beyond where we're at for now.

Thoughts?

This is incredible! I am trying to do this with my own music, however the 3d printer i have access to can only print an area 8in wide. I tried changing the processing code to reflect that and it failed. Any advice what needs to change to shrink the diameter of the 3d model? I really want to try this!

Thats exceptional...

Awesome! If you do a 7" 45, both the resolution could be higher (pickup moves past more groves per time) and the files smaller/easier to generate, right? Experimented with any 45 rpms?

Ah, of course the quality is lower at 7" than at the outer edges of a 12" anyway. But still interested in your progress, if you ever did try with RIAA eq, higher RPMs etc.

Ah, of course the quality is lower at 7" than at the outer edges of a 12" anyway. But still interested in your progress, if you ever did try with RIAA eq, higher RPMs etc.

I love learning new and interesting things ...
and this web is full of this.
Thank you very much for sharing knowledge and continue to do so

Is this possible to make on Ultimaker? Can I make the vinyl 5" and can I make muliple short tracks on it ? Sorry for all the questions but you got me all excited lol

@Amanda Ghassaei

Hello. Have you ever considered to use GPU computing? That might be helpful for sound processing on my opinion.

Incredible. A little info off topic- as far as I know Kurt Cobain hated Nevermind, the album which contains Smells Like Teen Spirit. He hated this song even more. The reasons he stated: the album sounded too perfect and clean, much unlike Nirvana's first album. He felt they had sold out. I wonder what he would say if he heard how Smess like teen spirit sounds on this record. :D

Learned some cool things in this tutorial, thank you.

This is such a cool program! But I am having trouble running the processing sketch. It says

"No library found for unlekker.util

No library found for unlekker.modelbuilder

No library found for ec.util

Libraries must be installed in a folder named 'libraries' inside the 'sketchbook' folder."

I tried putting the files in a bunch of different folders but non of them seemed to work. Any thoughts?

- I have the same trouble! Any way to fix it?

the modelbuilder library isn't in the right folder.

this project uses the same library, see if following the instructions from the top comment helps you:

https://www.instructables.com/id/3D-Printed-Photograph/

this is simply amazing! I can't believe it's printed!

- Hey guys,

I had an issue with the python script that kept sending me this error: IndexError: list index out of range

If you run into the same issue this fixed it:

Just add this line right below: for i in range(numframes):

if len(frameInt)>=(4*i+1):

Cheers,

Leart

What a creative technology 3D printing. Btw, can I made iPhone with 3D printing machine?

Great

This is incredible. I'm impressed at how far 3D printing has gone. Someday most of our items will be 3d printed, I assume. What do you think?

- Hi Amanda, I just sent you an e-mail regarding a film shoot inquiry. Can you please check your gmail and let me know what you think? Thank you~

Amazing!

- The problem seems to be in the nature of the printer output design. I think a much simpler solution would be to develop a printer that printed a long line for the audio then use a separate device that rolled it into a record like a really short but very thick jelly roll. I don't know if my idea makes sense but basically you'd be printing and then rolling a line of sound into a record format. I don't know of any type of printer that does this though or if one could even be made. It just seems that printing it all at once as a single object is where the problem with audio quality is coming from.

Thank you for all what they offer

- I have just showed this to my cousin and the reaction was ... @#!

deym!

- Hi, Amanda, great project. Can you tell me how can i make a sound loop record?

Smells like teen spirit is a timeless song. And that record is so cool. Great job.