Music Box Generation from MIDIs

6.S079 Final Project Report
Tiffany Lu and Trevor Walker

Motivation

The problem we are tackling with our final project is allowing users to generate custom music boxes by simply uploading a MIDI music file containing the melody. The key goals in this task are to decrease the cost to the user of creating a custom music box and to make it easier to generate boxes by allowing a user to upload the desired song instead of having to manually input each note and the rhythms while still maintaining good sound quality.

The custom 3D-printed music box was suggested in class, but there were two main problems with it. First, inputting the song for the music box was very clunky and involved manually specifying each note and when it should appear on the cylinder. Second, the fully 3D-printed box did not sound very good due to the plastic's thin sound. We plan to design a new system that accounts for both these problems by allowing users to upload MIDI files instead of manually inputting the notes, and by producing a vector file to cut the comb out of stainless steel rather than print it in plastic, for better sound quality.

Technical Approach

In order to generate the custom music boxes, we built a JavaScript web application that allows the user to upload a MIDI file and outputs .STL and .DXF files for 3D printing and laser cutting. There are four main components to our system: the JavaScript MIDI parser, the cylinder generation, the comb generation, and the physical box assembly. The web application provides an interface that allows users to upload a MIDI file and then subsequently view and download the generated files.

MIDI Parser

The role of the JavaScript MIDI parser is to extract the relevant note and timing data from the MIDI file to pass to both the cylinder and comb modules. The web application provides an input field for the user to upload a file, and once chosen, it is read as a binary string and fed into a JavaScript MIDI parser called jasmid [1]. The parser surfaces a "header" portion of the MIDI file, which contains the MIDI format, number of tracks, and the timing interval, as well as a "tracks" portion of the file, which contains the actual note data. The "tracks" property contains a list of events for each track. We use only the MIDI "note on" event to signify when a note should be played, ignoring the "note off" and other events because a music box cannot sustain notes or change modes.

To generate the comb, we require all unique notes that are played through the whole song, so we iterate through the "note on" events to gather the unique MIDI note numbers. To generate

the cylinder, we require [comb-tine number, timing] pairs. We use the list of unique notes to get the comb-tine numbers, and to get the timing, each event also has a delta-time property, which indicates the time between the current event and the last, so we build the [comb-tooth number, timing] pairs by accumulating those differences. These two arrays are passed on to their respective modules.

Comb

The comb component is a OpenJsCad [2] routine that takes in the list of unique notes in the song in the MIDI note number format and outputs a .DXF file for laser cutting. The lengths of the teeth on the comb are calculated by taking the MIDI note numbers and converting them to frequency, and using the following equation to calculate the length:

$$L = \sqrt{0.162 \frac{a}{f} \sqrt{\frac{Y}{d}}}$$

where a is thickness, f is frequency, Y is the Young's modulus, d is density, and L is the length. For the stainless steel the comb is cut out of, the constants are:

$a = 0.036$ in

$d = 0.289$ lbs/in^3

$Y = 2.85 \times 10^7$ ksi

Cylinder

The cylinder component is another OpenJsCad [2] routine, which takes in the [comb-tooth, timing] pairs as described above and generates a 3D-printable cylinder (.STL) with pins in the corresponding locations. The spacing between the pins both along and around the cylinder is variable and depends on the length of the song and the number of unique notes the song contains.

We calculate the spacing of the pins around the cylinder using the length of the song and the circumference of the cylinder, and we calculate the spacing of the pins along the cylinder using the number of unique notes the song contains. To place the pins on the cylinder, we iterate through the array of pairs and calculate the angle offsets around the cylinder depending on the timing.

The pseudocode for the pin placement around the cylinder can be seen below.

```
notes = [[tine-number, time],...];
hSpacing = 2*PI*radius/songLength; // horizontal spacing per time unit
vSpacing = height/numNotes; // vertical spacing for different notes

function placePins() {
  pins = [];
  for (i=0; i<notes.length; i++) {
    tineNumber = notes[0];
```

```
      time = notes[1];
      angleOffset = hSpacing*time/radius; // calculated using arc length

      // translate to edge of cylinder
      x = cos(angle)*radius;
      y = sin(angle)*radius;
      z = tineNumber*vSpacing;

      // apply transformations
      p = pin();
      p = p.rotateZ(angle);
      p = p.translate([x,y,z]);

      pins.push(p);
    }
    return pins;
  }
```

Music Box Assembly

The box component is designed statically as a CAD file, which provides slots to insert the cylinder and comb. The box uses a worm drive attached to a crank to turn the cylinder. Each part is designed to be press fit to the other parts, so that the parts are able to turn, and so different cylinders and combs can be interchanged for different songs.

To build the physical box, the static box parts and the cylinder are 3D printed, while the comb is laser-cut from stainless steel.

Results

The web application allows users to download the static music box parts in a ZIP, and to upload a MIDI file to generate a downloadable STL and DXF. Screenshots can be seen in Figure 1, and the actual application is hosted at https://tweilu.scripts.mit.edu/musicbox.

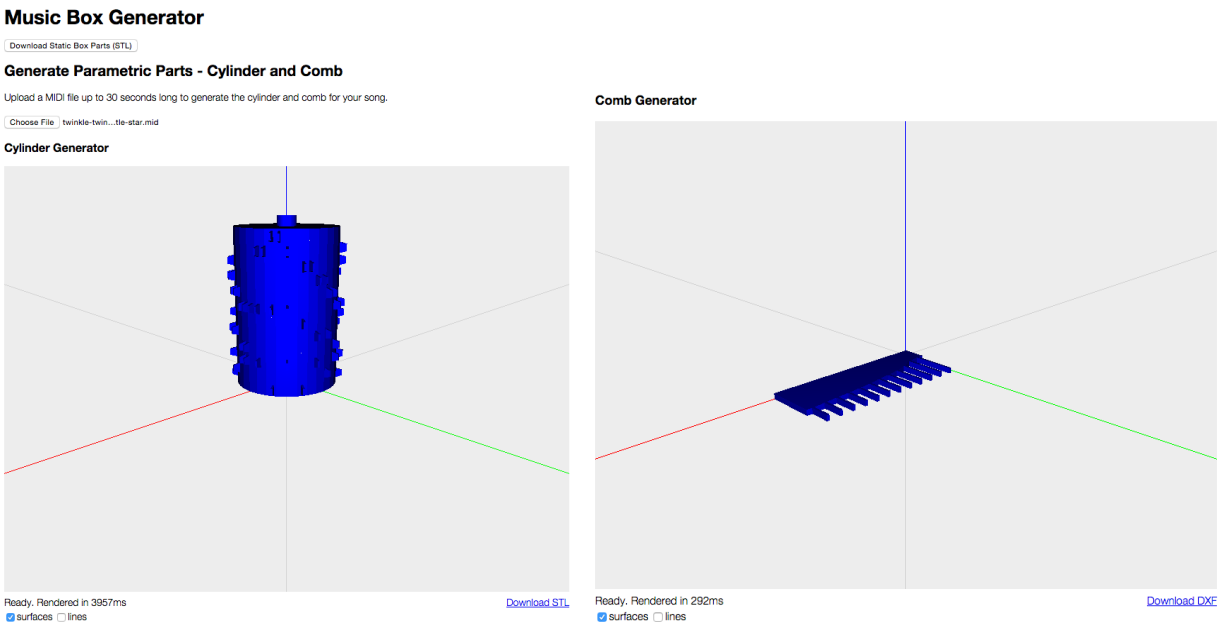

Figure 1. Screenshots of the web application's cylinder and comb generation.

A render of the static music box design is shown in Figure 2, and an example generated cylinder is shown in Figure 3.

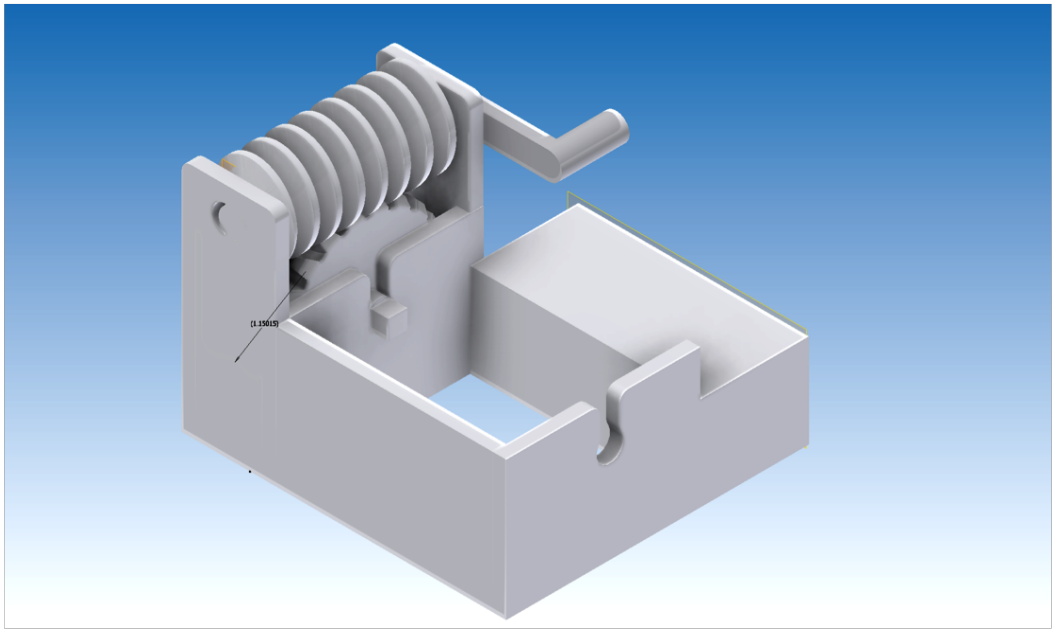

Figure 2. Static music box render.

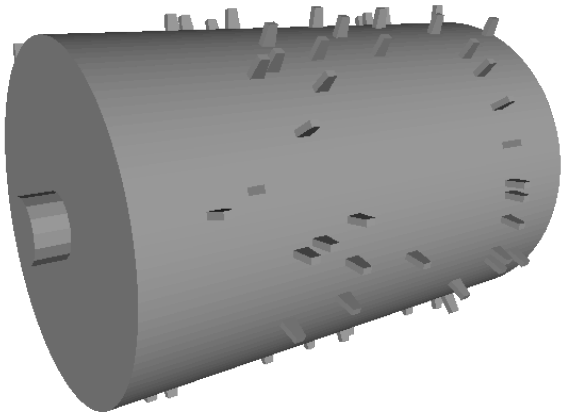

Figure 3. Parametrically generated cylinder for the Harry Potter theme.

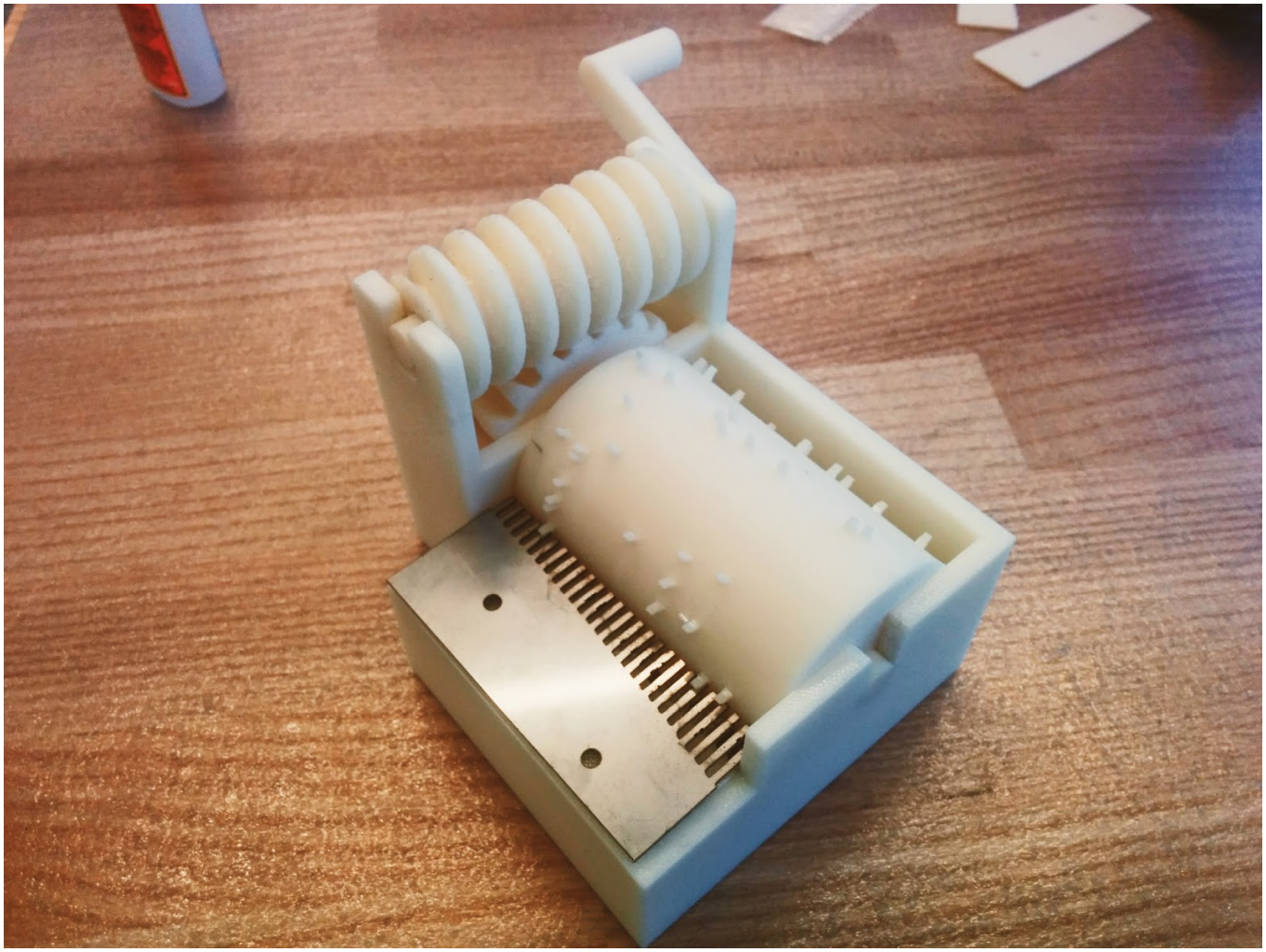

Figure 4. Final 3D printed music box with laser cut steel comb.

The final assembled music box can be seen in Figure 4. The box, box mechanisms, and cylinder were 3D printed on the Stratasys uPrint SE, and each print job took between 3 and 10 hours. The stainless steel comb was laser cut and ordered from Pololu [3]. However, the 1mm thick steel was too thick for the plastic pins on the cylinder to pluck without breaking, so we 3D printed combs at the last minute for the demo as seen in Figure 5.

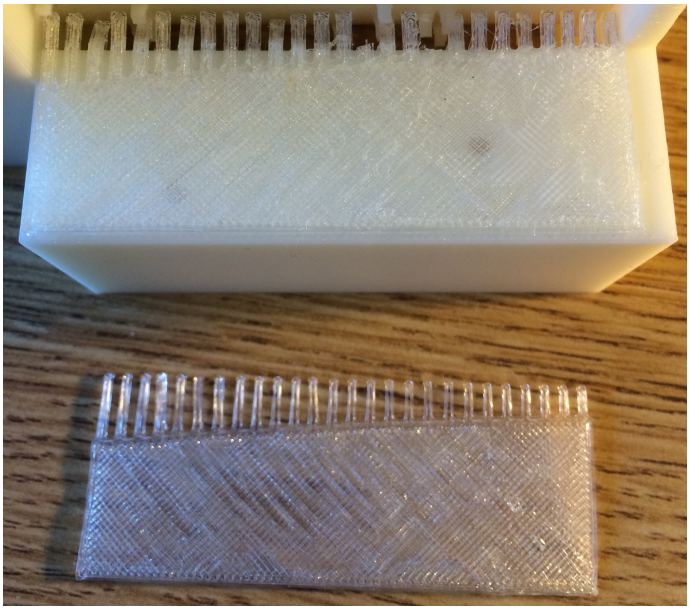

Figure 5. 3D printed combs for the Harry Potter theme.

A video of the music box in action with the plastic comb can be viewed at http://youtu.be/SXzM4Clmdk4. The video shows the lack of pitch associated with the plastic comb. The plastic comb is also a lot less durable than the steel comb. Figure 5 shows a few of the tines broke off.

Despite the steel thickness issue, we still tested the steel comb's pitch accuracy to verify our comb generation process. We recorded the frequency of each of the tines and compared it to the desired frequency. The results from this test can be seen in Table 1. Once adjusted for the octave difference, the error between the desired and measured frequencies is extremely low as demonstrated in both Table 1 and Figure 6.

Table 1: Frequency of comb teeth, compared with ideal values

MIDI note	Piano key	Fundamental frequency (Hz)	Measured peak frequency (Hz)	Harmonic-adjusted error (semitones)
41	F	87.30705786	1457	-0.3204487125
43	G	97.998859	1697	0.3193795372
48	C	130.8127827	2122	0.2381907287
52	E	164.8137785	2578	-0.391877982

55	G	195.997718	2115	-0.3321926099
57	A	220	3646	-0.4406845048
60	C	261.6255653	3662	-0.003582717419
62	D	293.6647679	4870	-0.4293979737
64	E	329.6275569	4922	-0.0786564001
65	F	349.2282314	4919	0.1052164497
67	G	391.995436	5550	0.1946890465
69	A	440	5696	-0.07279200839

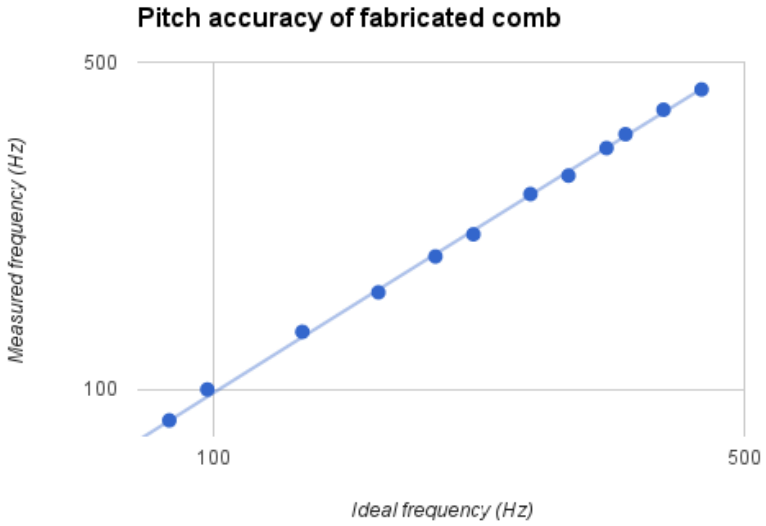

Figure 6. Plot of measured frequency (points) against the desired frequency (line).

Due to time constraints, we were unable to produce a second refined version of the music box. The 3D printing took more than a week because of failed MakerBot jobs and because our parts were fairly large. We ended up printing on the new Stratasys uPrint, which was much more reliable, but each job still took between 3 and 10 hours. Additionally, because we did not have access to a laser cutter that could cut steel, we ordered our custom combs from Pololu [3], and they took a week to arrive. So after we discovered that the 1mm thick steel, which works well with traditional music boxes, would not work with our printed box, there was not enough time to order another set of combs on thinner steel.

Potential Extensions

There are a number of possible extensions to this project to enhance the user experience. First, the steel comb would be instead cut out of, perhaps, 0.5 mm steel, so that the plastic pins on the cylinder are able to pluck them without breaking. This would be the biggest improvement for the music box.

Second, the music boxes are currently limited to approximately 30 seconds of music because if the song is any longer, the pins on the cylinder for fast repeated notes get too close together. This could be possibly solved in two ways to allow for longer songs. The radius of the cylinder could be adjusted to fit the song length such that the distance between notes is larger. This would require the box to become parametric as well as it would need to adjust to the size of the cylinder. The other option is to add in the possibility of having more than one tine of the same note on the comb such that repeated notes could be played on different tines, and the distance between the notes would no longer be a problem. The second option would likely be more favorable as it does not require the music box to change size.

Third, the process of 3D printing the music boxes for demonstration was the slowest part of the project. The parts were printed in separate jobs over the course of 4 to 5 days, some taking up to 10 hours. This could possibly be avoided by using alternative manufacturing processes for some of the components. For example, the static box model was the largest piece to print. The box could instead be laser cut out of acrylic, which would greatly reduce the total required printing time. The gears and cylinders would still need to be 3D printed, but this alternative would definitely speed up the actual music box assembly.

Finally, it could be interesting to provide more options to the user for generating and assembling their custom music boxes. While our project wanted the music boxes to use steel combs for increased sound quality, some users may want to just 3D print the whole thing, so the system could provide the option for the user to choose what material each component should be. The script to generate the plastic comb is already implemented because of the issue with the steel comb, so this extension would just require more options on the web interface. More unrelated, but still interesting, the application could also provide more input options to the system for which piece of music the box should play. For example, instead of requiring the user to upload a set length song, the interface could provide the user with a way to cut the song before generation. Another idea could be that the user could play notes in a virtual keyboard, and the system could output their own custom song.

Overall, the system could use some tweaks to make it more robust and give more flexibility to the user.

Conclusion

The resulting system provides users with an easy way to design custom music boxes that they can assemble themselves though there are still some interesting options available for giving the user more flexibility. The produced physical music box has low sound quality due to the plastic comb, but based on our analysis of the output frequencies produced by the steel comb, the sound quality of future iterations of the box would likely be comparable to that of traditional music boxes.

Tools & Packages

[1] Jasmid. https://github.com/gasman/jasmid.
[2] OpenJsCad. https://github.com/joostn/OpenJsCad.
[3] Pololu. https://www.pololu.com/.