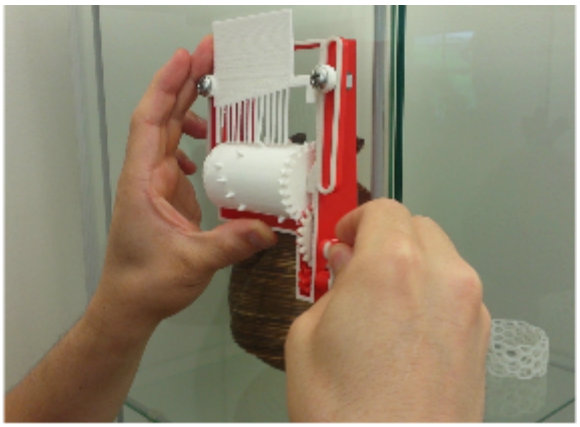
The Music Box Project

Lau, Moreno, Solanky

1. Motivation



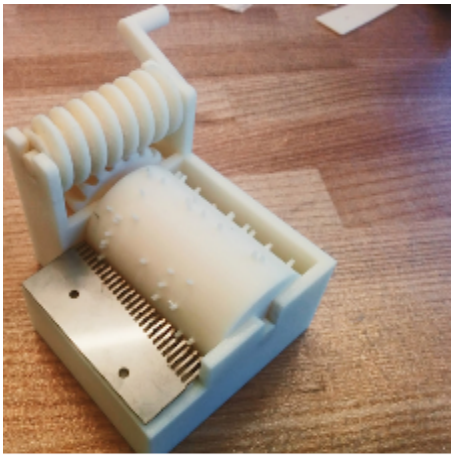
Our music box, with glass case amplification

Historically, humans have produced musical instruments from as wide a variety of materials and technologies as they have had access to. The profound power of computational fabrication lies in a greater facility for reproducing a work of imagination precisely in physical form. The ideal version of our project would enable a user to simply hum or whistle a tune into their microphone or take a picture of some sheet music, and automatically produce the design for a music box that would faithfully reproduce the user's input.

2. Related Work

Parametric Music Box by wizard23, found on Thingiverse

The resonant qualities of plastic are generally not suitable for musical instruments, which is why they are commonly made from metal or wood. However, working musical instruments have been made with 3D printers. Engineer Hans Fouche[4] and industrial designer Scott Summit[6] have both independently printed working guitars. The sound of string instruments and music boxes both work via the principle of forced vibration, indicating that the acoustics of 3D printed music boxes can sound good, even in plastic.



Lu and Walker's music box

Very little work has been done in the specific area of 3D printable music boxes. Philipp Tiefenbacher, aka wizard23, published a scad file for a working music box on Thingiverse[7], a website for users to share open source design files. Tiefenbacher's music box has several drawbacks. The comb and the case are produced as a single piece, which is inefficient for iteration and customization and damps the vibration of the comb unnecessarily. The herringbone gears have small crevices in which stuck support material and small errors in fabrication can produce noise and cause the gears to come out of alignment.

MIT students Tiffany Lu and Trevor Walker have also published code on GitHub for producing an entirely 3D printed music box.[3] Their design initially called for a laser cut steel comb, however the steel was not flexible enough and broke the plastic pins off the cylinder.

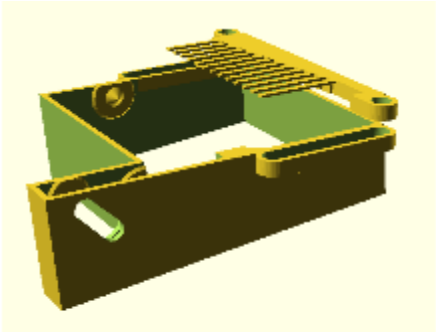
The plastic comb they used instead did not have good acoustic qualities. For the playing mechanism, they used a large worm gear that was both materially inefficient and difficult to use.

We decided to use Tiefenbacher's design as a starting point for several reasons. The design is compact, the code is relatively well structured for augmentation, and most importantly the comb produces a workable sound.

3. Technical Approach

A. Modularity

In order to tighten the iteration cycles in designing the musical comb and enable greater flexibility in customizing the arrangement of notes, we divided the comb and the case into separate modules. In order to fasten the modules together, we added bolt holes to the comb and corresponding slots on the case. Using this system, combs with different notes sequences can be fabricated and substituted for use with a single case.

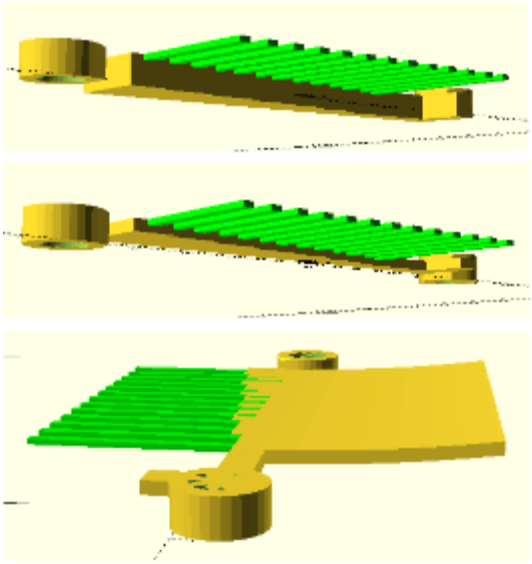


Modular comb with bolt hole and slot

B. Audio

I. Material Resonance

Taking inspiration from the soundboards of string instruments, we modified the now modular comb further by thinning and extending the body of the comb. Soundboards are thin so that they are more easily affected by forced vibration, and they have large surface area in order to vibrate a greater amount of air, thus amplifying the sound of the instrument's strings.



Top: Thick comb Middle: Thin comb
Bottom: Thin comb w/ soundboard

The main benefit for our music box is not in amplification of the sound, but in having a greater proportion of the music box's material vibrating at the right frequency. The soundboard is largely free of the damping forces of the rest of the music box. With the soundboard, a lesser amount of the energy propagated through the system via forced vibration is transferred into the resonant frequencies of the case, and instead is transferred to the soundboard, which vibrates more freely. This appears to have the effect of both clearer and longer sustained notes.

II. Pitch Accuracy

When plucked, the teeth of the comb oscillate at a frequency f determined by the length L and height h of the tooth, the stiffness E and density ρ of the printing material, and γ, the constant for the first mode of a cantilever beam.[2] Equation (1) describes this relationship as it is used in our code, as the length of each tooth is to be determined by the target frequency.

$$(1) \qquad L = \sqrt{\frac{\gamma^2 \cdot h}{4\pi \cdot f} \cdot \sqrt{\frac{E}{3\rho}}}$$

Unfortunately, the material constants E and ρ are unknown to us. Due to the variabilities inherent to the printing process, the stiffness and density may even differ between teeth on a single comb. Certainly there are minor variations in the dimensions of the teeth as well, due to difficulties in the removal of support structures.

In order to approximate the ratio $\frac{E}{3\rho}$, we first measured the frequency responses of the teeth of a test comb. With those frequencies and the dimensions of a set of thirteen teeth, we have thirteen sets of values for equation (2), which is a simple reformulation of (1). By averaging the resulting values for $\frac{E}{3\rho}$, we can approximate the correct material constants.

$$(2) \qquad \frac{E}{3\rho} = \left(\frac{L^2 \cdot 4\pi \cdot f}{\gamma^2 \cdot h}\right)^2$$

C. Song to Cylinder Mapping

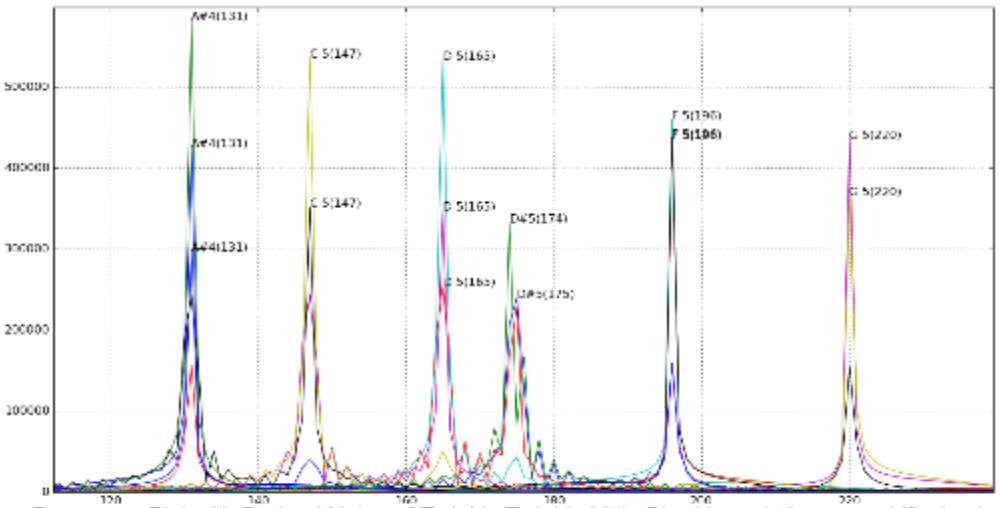
To allow for user specified songs to be be played with the music box, we experimented with a few approaches but ended up using two methods. Specifically we implemented methods to create an STL of the music cylinder with the required pin map from a MIDI, WAV format or recorded audio. The process involved dividing input sounds into chunks and extracting the dominant note using a frequency analysis of the input sound. This created the required note per time chunk format that was then passed to an SCAD file to create an STL music cylinder with the required pin mapping. We tested our system with synthetic tunes created in WAV format songs, specifically a 44KHz, 16-bit WAV file. These tunes were created using SonicPi[5], an algorithmic music application. We first attempted implementing the FFT analysis of the sound by passing microphone and sound recording through a custom GnuRadio[2] pipeline. This provided us real-time waterfall images of the frequency response of the input sound. Extending this pipeline to get notes per time chunk proved difficult due to its visual programming paradigm. Thus we shifted to a purely python based implementation for our program. While we tried other input mechanisms that could be implemented in future work like sheet music, ABC notation[8] to tunes. The method used required the least effort given the time-constraints. The programming methodology involved lots of empirical analysis of what kind of frequency responses were being produced by different notes, sampling rates and tune length. After much

experimentation involving data collection and analysis of variation in extracted note versus input note frequencies for different input formats. We derived a pitch multiplier magic number (3) for a 44KHz, 16-bit and 5 second tune in WAV format divided into 35 time slots.

$$(3)\ Pitch\ Multiplier\ =\ 12.2$$

With this multiplier we were able to extract the correct notes for all input sounds with the above constraints.

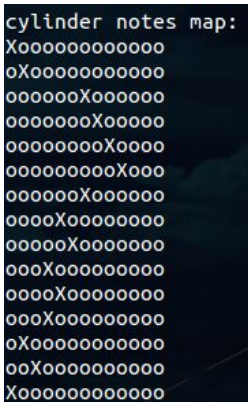
The pitch data was normalized according to (4) to make it independent of the sampling rate, duration of the WAV file and the time slots available on the cylinder, which decided the number of chunks a tune was to be chopped into.



*Frequency Plot with Derived Notes of Twinkle Twinkle Little Star *(annotations are shifted up)*

$$(4)\ Pitch\ Normalizer\ = (\frac{data\ length}{229376} \times \frac{sampling\ rate}{44000} \times \frac{time\ slots}{36})^{-1}$$

D. Notes to Cylinder Mapping



The SCAD file can take take a mapping of pins on the music cylinder in a serialized **X(pin)** and **o(no-pin)** format. This pin mapping format is created by our tune to cylinder mapping program and passed as arguments for the SCAD file to create an STL file. The whole pipeline is automated from the input tune provided by the user to the output STL file created by the program. The pin mapping is visualized in the figure on the right.

We also created an optimization to increase time slots available on a given cylinder for tunes using less than half an octave of notes. If a tune used fewer than six notes, we double the positions available on cylinder for each note and duplicate each note teeth. This allowed us to add more time slots to the cylinder as consecutive notes were not put at the same teeth. Thus the distance between time slots, which was previously constrained by the rotation required to pluck a teeth, can be reduced. This effectively allowed us to increase the granularity or length of the tune playable on the music box while keeping the cylinder diameter constant for simple tunes.

E. A second song to cylinder approach

As noted above, the SCAD file takes a string of **X(pin)** and **o(no-pin)** format in order to determine where to place the notes on the cylinder. To generate this mapping we also implemented a system whereby a user can pass a midi file into the system and generate a pin mapping and the corresponding notes.

The midi file format essentially consists of *tracks* which are represented as arrays of *events*. Each midi event then contains the meta-data for said event. For example a NoteOn event contains the tick (relative to the previous tick) that the note starts on as well as the integer representation of the note that should be played on that tick, while a setTempoEvent will contain the tempo of the song.

Using these events it is relatively simple to go from a midi file to the note mapping that we need. First we get the track that we are going to transcribe and change the ticks to be absolutely positioned in time instead of relative to the previous tick (this makes it easier to map on the cylinder). We then get the unique notes by looping through the track events and adding the pitch names to a set. From this, we sort the notes based on musical notation starting from *C4*. This is an arbitrary start note for our testing purposes but it does not matter so long as the notes are sorted from lowest to highest. Once we have this we simply loop through the events in the track and add the event to the final string based off of the time that the tick occurs and then output the string along with the ordered list of notes for use in our SCAD file.

Limitations

Currently the timing is off for certain songs due to the nature of midi files. Midi is a very forgiving format and so a lot of midi files from the web are missing crucial parts such as the song tempo or tick time.

4. Results

In order to evaluate the sound of the music box, we used applied the fourier transform to the sound of individual teeth, and plotted the results against the background of the array of target frequencies. At first, the largest discernible peaks in the frequencies were clumped together, with no real differentiation between separate teeth. These frequencies, consistent from tooth to tooth, are presumed to be the resonating frequencies of the comb. Accordingly, the plot changed significantly as we reshaped the comb to resemble a soundboard, whereby different teeth came to have more distinct frequencies, overcoming the inherent noise of the system.

With the estimates of E and ρ provided by Tiefenbacher, we found the resulting frequencies were about three octaves higher than predicted. Due to the material interference described above, it was difficult to determine which of the measured frequencies corresponded to the predicted frequency of (1), so it took several iterations of evaluating equation (2) for the teeth with distinct responses, averaging the results, and

updating the model before converging on $\frac{E}{3\rho} \approx 835560$

.

Even with this improved estimate, the resulting teeth do not perfectly match their target frequencies. In order to compensate for these unavoidable errors, we can extend the tooth length by a small percentage, which will ensure the resulting frequency will be slightly lower than the target frequency, and then sand down the sides of the tooth until the frequency is raised to the target.

On the tune to cylinder mapping front, after evaluating multiple mechanisms we ended up using two input formats: WAV and MIDI to extract the dominant note sequence and cylinder pin mapping to faithfully reproduce the tune. For the WAV input format we had to make the system independent of the available timeslots on the cylinder, sampling frequency and length of the tune using the normalization factor (4).

Top: Frequency response of original comb Bottom: Final comb

We created a working prototype that put together the comb and the cylinder programs to test our tune reconstruction. Qualitative testing suggested that we were able to recreate the desired tune but with ample room for improvement in audio quality.

5. Limitations

In order for the frequency of vibration to be perceptible, the teeth need to be quite thin. Unfortunately, the printing material is fragile at the required thinness, and the teeth can easily break while removing support structures or during use. The quality of the sound is still quite dull, as plastic is not very resonant. In addition to the fundamental inconsistencies in the teeth, the mechanics of the music box are affected by small errors, which can produce noise and malfunctions. The length of a song that can be faithfully reproduced is limited by the circumference of the cylinder and the required displacement of teeth for a good sounding pluck. Our input method was tested only with simple synthetic sounds from WAV files, not the common lossy and multi-tonal files that users are likely to have on hand.

6. Conclusions & Future Work

After much experimentation with the comb geometry and notes to cylinder mapping. We ended up creating a rudimentary but functional, tune to custom music box pipeline that we had proposed at the start of the project. The current mechanism can take in simple user input synthetic tunes and output a modular and fully 3D printable music box. Future work would involve improving the acoustic qualities of our system with a more holistic analysis of the comb and case resonance, e.g. with the finite element method. Further improvements like a graphical front-end and the ability to deal with more involved tunes may help ease usability and practicality of our software. More generally, computational fabrication opens up new avenues for designing simple objects, and our current music box design is still beholden to older fabrication processes. The comb inherently interferes with the acoustics, the length of songs are limited by the cylinder, so combless, cylinderless solutions should be explored. Combs and cylinders are after all just composites of extremely basic geometric shapes, and it is entirely likely that non-obvious topologies could sufficiently address these limitations. While using non-plastic material like metals could have improved the acoustics of our music box, we wanted to create a fully 3D printable music box to allow a wider set of users the ability to create their very own music boxes.

References

1. Durchschlagende Zunge. (n.d.). Retrieved June 03, 2016, from https://de.wikipedia.org/wiki/Durchschlagende_Zunge#Berechnung_der_Tonh.C3.B6he
2. Home page - GNU Radio. (n.d.). Retrieved June 03, 2016, from http://gnuradio.org/
3. Lu, T., & Walker, T. (n.d.). Music Box Generation from MIDIs. Retrieved from https://github.com/tweilu/6.s079-musicbox/blob/master/Documentation/6.S079FinalReport.pdf
4. Scott, C. (2015). The Remarkable Hans Fouche Returns with a 3D Printed Acoustic Guitar. Retrieved June 03, 2016, from https://3dprint.com/110959/hans-fouche-3d-printed-guitar/
5. Sonic Pi. (n.d.). Retrieved June 03, 2016, from http://sonic-pi.net/
6. Summit, S. (n.d.). 3D PRINTED ACOUSTIC GUITAR. Retrieved June 03, 2016, from http://www.summitid.com/#/island/
7. Tiefenbacher, P. (n.d.). Parametric Music Box by wizard23. Retrieved June 03, 2016, from http://www.thingiverse.com/thing:53235
8. Abc notation blog. (n.d.). Retrieved June 03, 2016, from http://abcnotation.com/blog/2010/01/31/how-to-understand-abc-the-basics/