Final Project

Drum Visualization & Animation

Alexander J Stathis
CMPS 161 - Animation & Visualization
University of California, Santa Cruz
astathis@ucsc.edu

Abstract

This project intends to provide a visual interpretation of the vibrations of a damped circular membrane (drum) after being struck. By applying solutions to the damped wave equation in two variables to produce a height value, the user can visualize the 'sound' made by a drum as waves rippling through the surface of the membrane.

1 Introduction

When a drum is struck, the surface ripples and displaces the air around it. The air around the drum is displaced in a manner similar to the ripple of the drum, which transmits waves through the air. The waves are transmitted to the ear, which is the sound that one hears. It is difficult to visualize the waves that one hears, but the waves that are transmitted through the drum are governed by a specific set of equations. By understanding the equations that govern the movement of the waves across the membrane, I hope to visualize these waves and in turn provide insight into the waves that travel through the air and produce the sound that one hears when a drum is struck.

2 Method

The drumhead can be modeled as a damped circular membrane in 2 dimensions. Because the circular membrane is highly symmetric, it is possible to parameterize the membrane by r and θ to take advantage of these symmetries. The following parameterization is used:

$$x = r \cdot \sin(\theta)$$

$$y = r \cdot \cos(\theta)$$

$$z = u(r, \theta, t)$$

where $0 \leq \theta \leq 2\pi$, $0 \leq r \leq R$, and z is the height of the membrane as explained later. R is the desired radius of the membrane, and without loss I choose R to be 1, as I just scale later when I go to draw the membrane.

Because I intend to provide an accurate representation of the waves that travel across the membrane, I must specify several restraints on $u(r, \theta, t)$. First is the condition that the boundary of the drum must remain stationary at all times. Second, the initial agitation of the drum is caused by a striking motion. In the instant the drum is struck, it is distorted heavily around a single point, while the rest of the membrane remains stationary. Essentially, for a small radius around the point of striking, the drum is drastically displaced in a continuous manner, but the rest of the drum remains flat. Also, because the drum resists such a striking, the instant that the contact between the stick and the membrane is released, the drum is initially at rest everywhere. The following equations model these conditions[4]:

$$u(R, \theta, t) = 0 \tag{1}$$

$$u(r, \theta, 0) = \frac{A}{2\pi\sigma} e^{-\frac{1}{2\sigma^2}[(r\cos\theta - r_*\cos\theta_*)^2 + ((r\sin\theta - r_*\sin\theta_*)^2]} \tag{2}$$

$$\frac{\partial}{\partial t} u(r, \theta, t) = 0 \tag{3}$$

In equation (2), A is the initial displacement amplitude, σ is the radius of the strike, and r_* and θ_* are the striking point of the membrane. To put it simply, as the distance from the striking point gets further, the amplitude of the displacement decreases exponentially. The exponential function guarantees the continuity condition of the strike, and σ can be varied to choose how local the displacement is.

The last restraint is that as the displacement transfers throughout the surface, it travels in a wave like motion. As such, I dictate that $u(r, \theta, t)$ must be modeled by the wave equation. I also require that this equation be damped, as otherwise our surface would oscillate forever. The wave equation in r and θ is as follows[4]:

$$\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2} = \frac{1}{c^2}\left(\frac{\partial^2 u}{\partial t^2} + 2a\frac{\partial u}{\partial t}\right)$$

where a is the damping factor, and c is the speed of the wave on the membrane.

Kin provides a solution to such an equation with the given boundary and initial conditions. He solves using separation of variables and the orthogonality relationship of sine, cosine, and bessel functions[4]. This solution is the height function I will use to model my drumhead.

$$\begin{aligned} u(r, \theta, t) = \frac{1}{2}\sum_{n=1}^{\infty}\left[J_0(k_{0n}r)a_{0n}\cos(\sqrt{\omega_{mn}^2 - a^2}\cdot t)e^{-at}\right] \\ + \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\left[J_m(k_{mn}r)(a_{mn}\cos(m\theta) + b_{mn}\sin(m\theta))\cos(\sqrt{\omega_{mn}^2 - a^2}\cdot t)e^{-at}\right] \end{aligned}$$

Where the constants a_{mn} and b_{mn} are given by

$$a_{mn} = \frac{2}{\pi[J_{m+1}(k_{mn})]^2} \int_0^1 \int_0^{2\pi} rJ_m(k_{mn}r)u(r,\theta,0)\cos(m\theta)d\theta dr$$

$$b_{mn} = \frac{2}{\pi[J_{m+1}(k_{mn})]^2} \int_0^1 \int_0^{2\pi} rJ_m(k_{mn}r)u(r,\theta,0)\sin(m\theta)d\theta dr$$

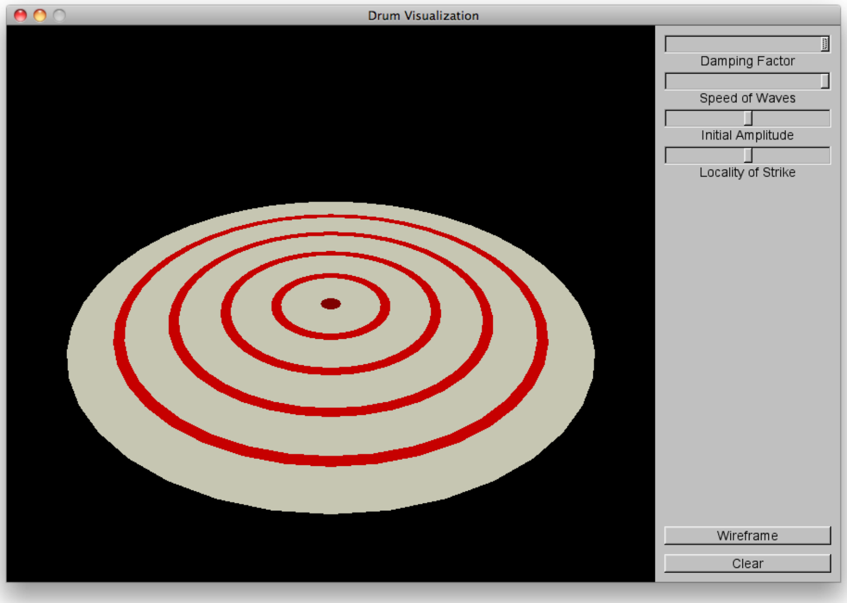
and $\omega_{mn} = ck_{mn}$, J_m is the m^{th} Bessel function, and k_{mn} is the n^{th} root of J_m.

Each zero of the Bessel functions is a different mode of vibration of the drum. Zeros of the 0^{th} Bessel are the symmetric modes (displacement is the same regardless of θ) and zeros of the m^{th} Bessel function for $m \geq 1$ are asymmetric modes of vibration[1].

3 Results

The project is programmed using the C++ language with the OpenGL and FLTK (Fast Lighting Toolkit) APIs. The actual drumhead is drawn with the OpenGL API, while FLTK is used to create the windows, handle user intactivity (both in presenting and handling GUI interaction and when the user interacts directly with the OpenGL window), and telling the OpenGL window to redraw.

The OpenGL code is contained in a class derived from one provided in the FLTK API, and displayed in a sub window of the main FLTK window. This class contains a static timer method which ticks every $\frac{1}{24}^{\text{th}}$ of a second and causes a redraw of the OpenGL code. Each time a redraw is done, the circle is drawn using r and θ coordinates which are converted to x and y coordinates via the parameterization provided in *Method*. A predefined hardcoded θ-resolution and r-resolution are used to draw the circle. For instance, the 0 to 1 range is divided into r-resolution sections, each of which contains a vertex for every θ-resolution. A quad is drawn using the n^{th} and $(n+1)^{\text{st}}$ r-sections and θ and $\theta + \theta$-resolution points as vertices. For each such vertex, a height is computed using the technique outlined below, but initially the membrane is at rest.



The drumhead initially at rest

3.1 Computing Heights

3.1.1 Computing a_{mn} and b_{mn}

Each time the user interacts with the membrane (agitates the drum in a different position), the coefficients a_{mn} and b_{mn} are recalculated. As these are double integrals, I computed them numerically using the Trapezoidal Rule. The Trapzeoidal Rule is as follows[7]:

$$\int_a^b f(x) \approx \frac{b-a}{N}\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{N-1} f(a+k\frac{b-a}{N})\right]$$

where N is the number of 'steps' (a larger number of 'steps' results in a closer approximation) used. Due to the fact that these only needed to be recalculated once when the user interacts with the membrane, these can be calculated using a reasonably large N, thus making these approximations relatively accurate.

3.1.2 Computing $u(r, \theta, t)$

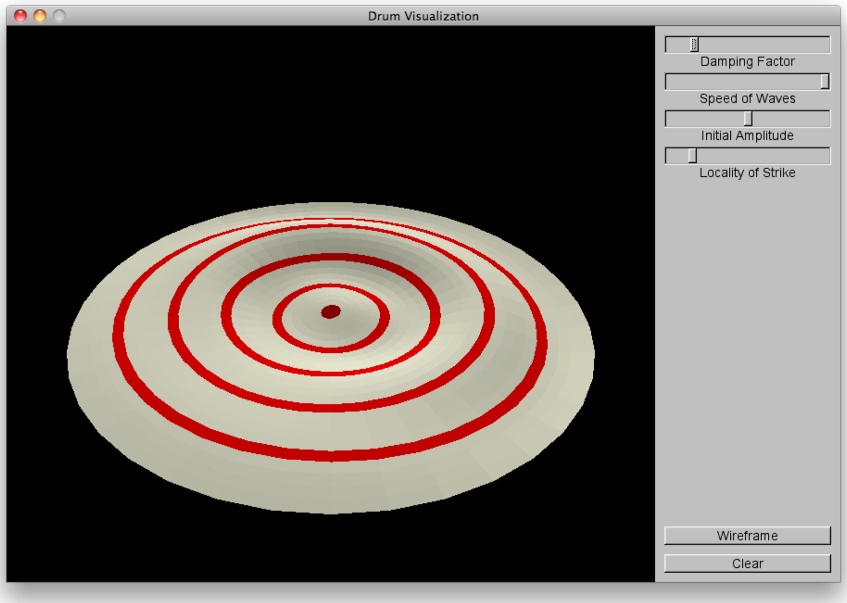
Due to the infinite sums in the solution to height function, the height at each point is only an approximation. This approximation is in fact quite rough, as only the first 4 symmetric and asymmetric modes are summed. Summing further proved to be extremely costly, as it requires two more computations of the Bessel function for each vertex in the resolution (roughly 2000 extra computations per extra sum per redraw).

The Bessel zeros are calculated initially when the program begins running. The Bessel function and its zeros are calculated using the GSL (GNU Scientific Library) API. Originally, I had decided to approximate these via one of the definitions[2] of the function itself, but this proved too costly and inaccurate.

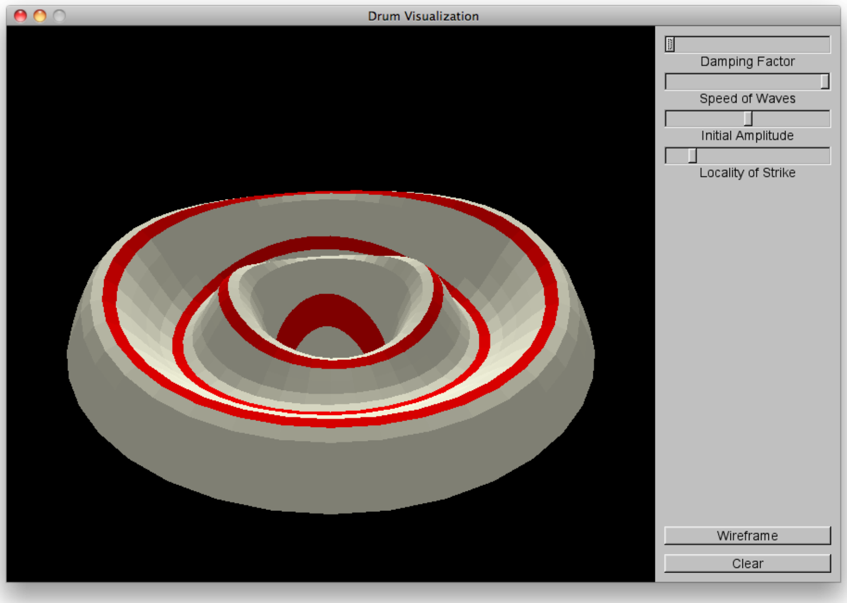
3.2 Controlling the Waves

3.2.1 The GUI

Using the FLTK API makes it relatively easy to implement user controls. Sliders are used to control the value of the damping factor, the speed of the wave on the membrane, the initial displacement amplitude, and the locality of the strike.



The drumhead after several seconds with some damping

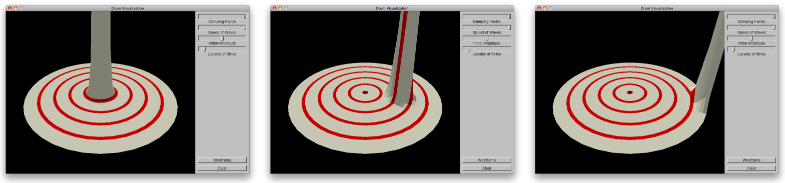


The drumhead after several seconds with no damping

Buttons are implemented similarly to both reset the membrane back to its resting position and also to swap back and forth between wireframe and solid viewing modes.

3.2.2 Interacting With the Membrane

FLTK's native window handler is used to capture mouse click coordinates. These coordinates are then converted into OpenGL space coordinates using the inverse projection and modelview matrices via OpenGL's `gluUnProject` function. I then apply the inverse parameterization to obtain the coordinates of r_* and θ_*, and recalculate the coefficients as described above in *3.1.1*.



Three images depicting different initial striking locations

4 Conclusion

In general, this project provides exactly what it set out to do: a very solid user interactive visualization of the vibrating drum. User interactivity is fairly robust, as many of the constants are controlled via sliders. The control is in depth enough that the membrane can be displaced in a variety of ways.

This project leaves a lot of room for implementation of extra features, which will be outlined below in the *Future Work* section, but provides the basic functionality that was originally intended.

5 Future Work

1 Extra User Interactivity

Possible variables to control include the tension of the drumhead and the density of the drumhead (or allowing different 'materials' to be selected for the drumhead). The same effects can be gotten via the damping and wave speed controls already available, but if *2* is to be implemented these new controls could more intuitively control the sound of the wave. Also, providing controls for different initial conditions for the drumhead (agitating the drumhead in different ways).

2 Sound

The height function outlined in the *Method* section above may also be used to generate sound. Frequencies and amplitudes for the vibrations of the sound will need to be computed some how, possibly via the methods suggested by Kin [4]. Extra libraries will need to be found and compiled in the project to provide such functionality.

3 Drum & Drumstick Modeling

It might be nice to enclose the drumhead in a nice drum model, along with a drumstick that follows the mouse location on the screen. Also, a canned animation for the drumstick when striking the membrane might be nice to show the initial agitation of the drum.

4 Different Drum Shapes

This implementation provides only a visualization of a circular drum head due to the fact that the height is computed using a closed form solution based on boundary conditions that require the symmetry of the circle. It should be possible to switch to an iterative method that computes heights based on previous values using integration, at which point it may be possible to implement different shapes, as long as the same restrictions on $u(r, \theta, t)$ are considered.

6 Related Work

Static videos of the vibrations of the drum are widely available throughout the web. These videos provide a nice representation of the vibrations of the drum, but they lack the the user interactivity and variability of the program outlined above.

Kin [4] provides small clips of drums being struck along a single radial axis at different displacements. These clips contain sound and are governed by the same equations provided above in the *Method* section.

Russell [3] provides several static videos of the some symmetric and non-symmetric modal vibrations of the drum. These are nice visualizations of the individual modal agitations, but do not provide interactivity or understanding of the actual vibrations of the drum when struck, just a several nodes.

Falstad [5] provides a similar representation to mine. He allows the user to interactively strike the membrane, and also has a nice visualization of which modes are vibrating after displacing the drum. He also generates sounds based on several variables of the drum membrane.

References

[1] Steve Zelditch. *Vibrating Drum.* Johns Hopkins University, F2005. `http://www.mathematics.jhu.edu/zelditch/Teaching/F2005110.302/PDF%20Lectures/DrumundBessel.pdf`

[2] Weisstein, Eric W. "Bessel Function of the First Kind." From *MathWorld.* `http://mathworld.wolfram.com/BesselFunctionoftheFirstKind.html`

[3] Daniel A Russell. "Acoustics and Vibration Animations". Kettering University, Flint, MI. `http://paws.kettering.edu/~drussell/Demos/MembraneCircle/Circle.html`

[4] Hon Kin. "Sound Simulation of a Drumhead". The Hong Kong University of Science & Technology. `http://www.math.ust.hk/ machas/drum/`

[5] Paul Falstad. "Circular Membrane Applet". 25 March 2005. `http://www.falstad.com/circosc/`

[6] Wikipedia contributors. "Bessel Functions." *Wikipedia, The Free Encyclopedia.* 25 Aug. 2004. Web. 10 Mar. 2011. `http://en.wikipedia.org/wiki/Bessel_function`

[7] Wikipedia contributors. "Trapezoidal Rule." *Wikipedia, The Free Encyclopedia.* 20 Apr. 2009. Web. 10 Mar. 2011. `http://en.wikipedia.org/wiki/Trapezoidal_rule`