Auralizing Quantum Mechanics

A thesis submitted in partial fulfillment of the requirement
for the degree of Bachelor of Science in Physics
from the College of William & Mary in Virginia,

by

Breese Sherman

Keith Griffioen, Research advisor
David Armstrong, Dept. of Physics

Williamsburg, Virginia
May 1 2020

Contents

Abstract

Quantum mechanics isn't the easiest subject to comprehend. For someone to understand its foundations alone, they should have a strong grasp on multivariable calculus, complex numbers, and Euler's formula. Or should they?

Quantum mechanical behavior, due to its similarities to the physics of sound, can be in part expressed through visualization, and more importantly auralization, of combinations of sound waves. Our project this year was to build a tool that allows any user, regardless of background knowledge, to easily do so.

The tool is a web app, so it can be accessed from any platform. The user can choose to "watch and hear" the behavior of a quantum particle inside a number of different potential wells, including those that simulate the quantum harmonic oscillator and the hydrogen atom. Over the course of the year, we have added feature after feature to this app, such as the ability to select multiple eigenstates at once (creating a superposition — or in this case a chord), the option to select a certain output waveform for better audibility, and the ability to change various parameters of the potential well itself.

We hope that through the use of this app, the world of quantum mechanics might be more easily accessible to students of all ages and backgrounds.

Chapter 1

Introduction

1.1 Interactive Education

On one level or another, everyone has experienced the joy of interactive learning as early as preschool. Over and over, it has been proven that there is no other way to get through to a young person's head as easily or efficiently. But as the student ages, these methods of active learning become less and less relevant to their education. Though the human brain comes pre-built with several different "passageways" for various forms of information to enter [1], the method of verbal teaching is ultimately ubiquitous across all institutions of higher education. The pervasiveness of the traditional lecture-based form is rooted in its benefits: it's of lower cost than, say, providing each student with an expensive piece of equipment, as is the case in hands-on "lab" classes, and it's much easier for an instructor to deliver information to a group, rather than interact with each student one-on-one. However, there are instances where the drawbacks of this method outweigh its benefits, particularly when verbal instruction simply doesn't communicate a given subject effectively enough.

In *Multimedia Learning*, Richard Mayer asserts that no matter how coherent and well-written it may be, a 500-word passage about how lightning storms work can still be insufficient for a learner to commit its contained information to memory [2].

Lightning storms are inherently visual phenomena — a block of text describing in excruciating detail how electrons move from the base of a cloud to the ground forgoes the striking, unforgettable imagery of a lightning strike in favor of a mathematical explanation. Similarly, when a typical quantum mechanics student sees the generalized wave function for the quantum harmonic oscillator potential,

$$\psi_n(x) = \frac{1}{\sqrt{2^n n!}} \sqrt[4]{\frac{m\omega}{\pi\hbar}} e^{-\frac{m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}} x\right) \qquad ,$$

for the first time, that student sees nothing but a complicated equation, and completely misses out on its mentally realizable, *human* qualities — qualities that can be brought to light through visualization (as many paper textbooks have done before) or in our case, auralization.

I should make clear here that I will be using the term "auralization" loosely. Rather than its strict dictionary definition, I am taking it to mean the aural analog of visualization — that is, taking information that isn't aural in itself, and converting to auditory information by any means.

1.1.1 Auralization

Human ears are finely tuned machines that can sense very slight changes in acoustic environments. A perfect fifth (seven semitones) sounds much more harmonious than a tri-tone interval (six semitones), even though they may be only a few hertz apart from each other. We'd like to use this to our advantage when presenting quantum mechanics in a different light — amplifying the nuanced differences between the behavior of particles in different potential wells.

1.2 Objectives

Our end goal is to be able to present quantum mechanics aurally in a way that makes sense to a student through the use of an interactive computer tool. To avoid over-loading the student with information, or not giving them enough information at any one time, we try first and foremost to abide by Mayer's "Principles for Reducing Extraneous Processing in Multimedia Learning" [2]. Using these principles as guidelines in writing our program will streamline its design into four separate goals that we can meet individually. Secondly, we want our tool's information and interactivity to be comprehensible by as wide a range of learners as possible, even those who know nothing about quantum mechanics. This is to say, we want the tool to be more than a noisemaker for someone who's never heard of quantum behavior: the tool should give them the information itself, rather than requiring them to read or even skim this paper. Lastly — and this is more or less a given — we want the non-educational aspects of our program to be straightforward and unambiguous; that is, we want a clean user experience.

1.2.1 Mayer's Principles

In his book *Multimedia Learning* (2001), Richard E. Mayer outlines four important rules that govern the degree to which multimedia education succeeds or fails [2]. Given that the program we are building is sensory in more ways than one, we try to abide by these rules as dutifully as possible. They are as follows: the Coherence Principle, the Signaling Principle, and the Spatial and Temporal Contiguity Principles.

Coherence

"People learn better when extraneous material is excluded rather than included."

— Richard E. Mayer, *Multimedia Learning*

To keep our tool simple, we'd like to keep the focus on the pure visuals and sound rather than the raw information about how a quantum particle behaves. That, should the user want to explore it, will be covered in here, in the next chapter.

Signaling

"People learn better when cues that highlight the organization of the essential material are added."

— Richard E. Mayer, *Multimedia Learning*

We'd like to make an effort to keep the behavior of the four potential wells distinct — and yet similar in the ways that matter. In this way, we'll make an effort to highlight the differences in their shapes and sounds while presenting all of them in the same way.

Spatial contiguity

"Students learn better when corresponding words and pictures are presented near rather than far from each other on the page or screen."

— Richard E. Mayer, *Multimedia Learning*

This will factor heavily into the design of our program — specifically, energy values for eigenstates will be displayed directly next to the eigenfunctions themselves, eigenfunctions will be displayed on top of the clickable "energy levels," so on and so forth.

Temporal contiguity

"Students learn better when corresponding words and pictures are presented simultaneously rather than successively."

— Richard E. Mayer, *Multimedia Learning*

Ideally, every bit of information necessary to understand a quantum particle's behavior will be presented at once when the user begins interacting with the program. This requires that all stationary states' eigenfunctions be displayed at the same time as their superposition, as well as their respective energies.

1.2.2 Accessibility

We would like this tool to be accessed from anywhere by anyone, easily. This will most likely mean uploading it to a website so it can be accessed with a simple link. This requires a website, which luckily already exists. My personal site `https://itsbreese.com` runs on a remote Google Cloud server that can host just about anything we put on it, so we will store it there.

Chapter 2

Goals

2.1 A Brief Overview of Quantum Mechanics

In order to understand quantum mechanical eigenstates, we must first examine the behavior of a classical particle when confined to a certain area. Specifically, when a spherical ball (perfectly spherical in this case) is placed between two immovable borders and given some arbitrary momentum, the ball will bounce back and forth between those two borders for eternity. This is because, in our perfect world, the ball does not lose any energy over time, by Newton's first law. But as we know, the world is not perfect, and the ball will slow down eventually due to energy lost from friction or the action of bouncing off the walls. Our example breaks down further as we delve into the world of quantum mechanics, where this ball and its enclosing box (called a "potential" become very, very small — on the order of the width of a hydrogen atom. In situations such as these, "Schrodinger's equation" is more suitable to predict the ball's position and momentum over time.

$$i\hbar\frac{\partial\Psi}{\partial t} = \frac{-\hbar^2}{2m}\frac{\partial^2\Psi}{\partial x^2} + V(x)\Psi$$

In this equation, Ψ represents not the particle's position, nor its momentum, but the complex probability amplitude that the particle might be in a specific place at a specific time. This uncertainty arises due to Heisenberg's uncertainty principle, which

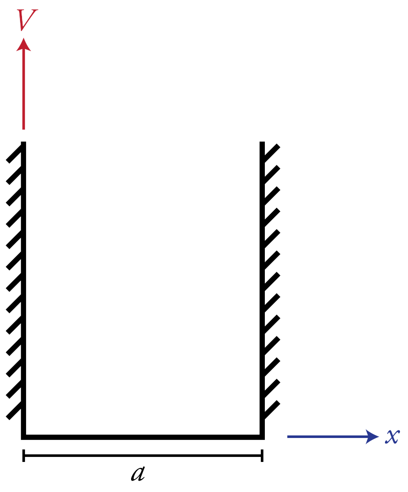

Figure 2.1: A depiction of the "infinite square well" quantum potential. Beyond the "walls", V is equal to ∞.

states that the product of the standard deviations of a quantum particle's position and momentum must always be above a certain value, that is, $\sigma_x\sigma_p \geq \hbar/2$, with $\hbar$ being the reduced form of Planck's constant. Also, $V(x)$ represents the shape of the "potential energy well" that the particle is inside — in the case of a particle in a perfect box, as above, $V(x)$ would look as it does in Fig. 2.1, where V is equal to zero in a region of width a, and infinity everywhere else.

The precedence that Schrodinger's equation has over quantum mechanics allows for some very interesting results to arise, but first we must look deeper into what $\Psi(x,t)$ is. It is called the *wavefunction* of the particle, and it is usually complex, meaning that it must be multiplied by its own complex conjugate to return the probability of the particle being in a location x at time t. Another thing to note about it is that due to the nature of Schrodinger's equation, it must always meet certain conditions specified by $V(x)$, the shape of the potential well. This property, combined with its wave-like nature (note the Schrodinger equation's resemblance to the classical wave equation), allows some very interesting mechanics to arise — namely, that the wavefunction has to be one or a linear combination of discrete solutions, which we call *eigenstates*. These states are usually referred to with a lowercase ψ and do not change with time ($\psi(x)$), unlike $\Psi(x,t)$.

2.2 Eigenstates

Solving Schrodinger's equation for a particle in an arbitrary potential will always give a discrete set of solutions for Ψ (though that set may be infinite). In most cases, $\Psi(x,t)$ can be broken up into a separable sum:

$$\sum_n C_n \psi_n(x) \cdot e^{\frac{-iE_n t}{\hbar}},$$

where we are summing over an arbitrary number of eigenstates ψ_n. Since these eigenstates do not depend on t (and therefore do not change with time), we call them *stationary states*. But what creates the form of these different stationary states? That would be E_n, the energy of our particle — and remember that E_n is quantized, meaning that it can't be just any value. Specifically, in the case of our "perfect box," which, in the world of quantum mechanics, is termed the "infinite square well" (even though squares have nothing to do with it), the energy E_n of a particular eigenstate ψ_n is given by

$$\frac{n^2\pi^2\hbar^2}{2ma^2},$$

where n can be any natural number, m is the particle's mass, and a is the width of the well (box).

Let's say then, for simplicity, that $\pi^2\hbar^2/2ma^2$ comes out to 1 eV of energy, and so E_n has to be 1 eV, 4 eV, 9 eV, 16 eV, et cetera. How then, would a particle with 8 eV of energy be possible at all? Surely it must be feasible that a particle bouncing back and forth with 9 eV of energy suddenly loses 1 eV for whatever reason. So how can it have a specific wavefunction Ψ when no single choice of ψ_n seems to yield the correct result? This is where our sum comes in: this particle can be in any combination of

ψ_n states, just so long as the squares of their coefficients C_n add up to one — this is so that the probability of the particle being found in the box is exactly one, no more and no less. Now we just have a simple algebra problem: given that our particle is in a combination of ψ_2 (4 eV) and ψ_3 (9 eV) that yields an energy of 8 eV — that is, $|C_2|^2 \cdot (4eV) + |C_3|^2 \cdot (9eV) = 8eV$), and $|C_2|^2 + |C^3|^2 = 1$, we just have to solve for C_2 and C_3. When a particle's wavefunction Ψ is made up of more than one stationary state ψ_n, we say that the particle is in a *superposition* of states.

2.3 Energy

So, we have it that a particle in an infinite square well (or any differently-shaped potential, for that matter) in a combination of stationary states ψ_n has a total wavefunction Ψ, and each stationary state has a determined energy E_n. Nothing that $\Psi = \sum C_n \psi_n \cdot e^{\frac{-iE_n t}{\hbar}}$, we see that E_n plays an interesting role in what Ψ ends up looking like. (We can't graph Ψ normally, since it's complex, but we can graph its squared modulus $\Psi^*\Psi$ to give us a fully real function). How does E_n affect the shape of Ψ? Looking at the exponential term on the end of the sum, we see that when we take its modulus squared, we get a term that oscillates with time (i.e. one that looks like $sin(kt - \delta)$. So the wavefunction Ψ (but *not* the eigenstates ψ_n) oscillates with time somehow, and the speed of that oscillation is dependent on E_n, the energy of any and all eigenstates that are part of Ψ. For example, if a particle is in a single eigenstate ψ_2, so its energy E_2 is 4 eV (for simplicity), then the full wavefunction Ψ becomes:

$$C_2 \psi_2(x) e^{\frac{-i \cdot 4\text{eV} \cdot t}{\hbar}},$$

where C_2 is a constant number that allows for

$$\int_{-\infty}^{\infty} |\Psi|^2 dx = 1,$$

that is, the particle has a 100 percent probability of being *somewhere.* So, on account of the exponential term, Ψ oscillates in the complex plane at a rate of 4 eV/$\hbar$, but this can be higher or lower depending on the energy of the particle.

This is where sound comes in: *anything that oscillates sinusoidally can be expressed as a pitch.* We can "listen" to this eigenstate ψ_2, and all other eigenstates, by mapping their oscillation rates onto an audible range of frequencies. Since energy is all relative, we can say that ψ_1 begins at 200 Hz and then proceed from there... Recall that E_n, in the case of an infinite square well, is given by $n^2\pi^2\hbar^2/2ma^2$, so it becomes a simple matter to calculate the given frequency for any eigenstate. Since all four of the potential wells in our program obey different equations when it comes to calculating the energy of their eigenstates, we will have different relationships between the audible pitches for each potential.

In addition, superpositions easily translate to a combination of tones. If a particle is partially in the ψ_1 state and the ψ_9 state, we don't hear a single frequency that corresponds to the sum of E_1 and E_9, but we instead hear *both* frequencies. This can lead to some interesting chordal combinations — combinations that are different between each potential well.

2.4 The Four Potentials

There are four common theoretical potential wells that we will be employing in our program. Ideally, we will be able to clearly *hear* the difference between them, when we listen to the relationships between their various eigenstates. They are:

- The infinite square well
- The quantum harmonic oscillator
- The delta function

- The hydrogen atom

These four "shapes" are all spaces in which we could drop a theoretical particle, and they all affect its wavefunction Ψ differently. The first three are largely theoretical, but the hydrogen atom is to be found just about everywhere in the universe... so you might say it has a bit of practical application.

The infinite square well

We've gone over much of this one already. As stated before, the energy E_n is given by $n^2\pi^2\hbar^2/2ma^2$ (with a being the width of the well and m being the mass of the particle), so it's a simple matter to translate that to audible pitches. But what if we want to see the eigenstates ψ_n themselves? What we haven't looked at is the actual equation for ψ_n in this situation. It is given by

$$\sqrt{\frac{2}{a}}\sin\frac{n\pi x}{a}.$$

This is just a sine wave, which can be easily depicted on screen, oscillating at a rate proportional to its energy, just like the actual wavefunction Ψ. In addition, should we want to show the potential well itself, its equation is given by $V(x) = 0$ when $0 < x < a$, and ∞ at all other points. (Refer to Fig. 2.1 above).

The quantum harmonic oscillator

Similarly to classical harmonic oscillators, we say that $V(x)$, in this case, is equal to $\frac{1}{2}m\omega^2x^2$, where ω is proportional to the square root of the "strength," k of the oscillator. After a bit of work with operators, we have it that $E_n = \hbar\omega(n + \frac{1}{2})$. Very simple — the audible frequencies will ascend in pitch at a constant rate as n increases. This is to say, the energy levels are equally spaced. Displaying the shape of the potential well is simple as well — it goes as x^2. Meanwhile, the eigenstates ψ_n

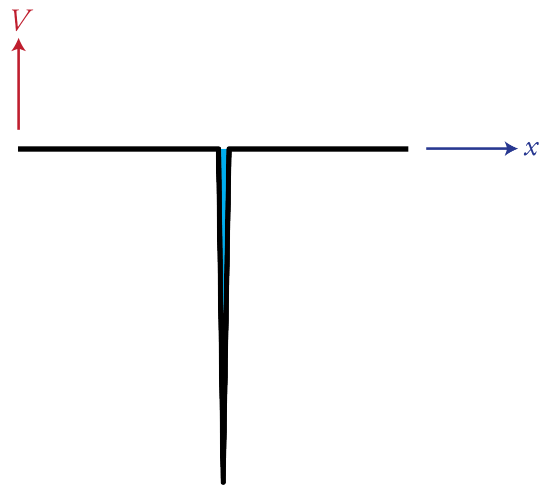

Figure 2.2: An exaggerated depiction of the Dirac delta function potential well. In "reality," the spike at the center is of zero width, and the blue shaded area has an area equal to $-\alpha$.

for this potential look like

$$\frac{1}{\sqrt{2^n n!}} \left(\frac{m\omega}{\pi\hbar}\right)^{1/4} e^{\frac{-m\omega x^2}{2\hbar}} H_n\left(\sqrt{\frac{m\omega}{\hbar}}x\right).$$

This equation may look frighteningly complicated, but it only depends on x, so we can graph it across the well just as we do in the previous case. H_n are the *Hermite polynomials*, which we'll have to hardcode into our program directly, as they depend on iterated n-th derivatives.

The delta function

In this interesting case, our potential $V(x)$ is equal to $-\alpha\delta(x)$, where $\delta(x)$ is known as the *Dirac delta function*. Simply put, $\delta(x)$ is equal to 0 at all points except $x = 0$ itself, where it has an infinitely sharp spike such that

$$\int_{-\infty}^{\infty} \delta(x) = 1.$$

An approximation of this idealized distribution is shown in Fig. 2.2.

This discontinuous function proves useful for quantum mechanics, where it becomes fairly simple to calculate Ψ and the corresponding energy levels for V. It just so happens — given that the total energy of the particle is less than zero (remember that energy is all relative) — that there is only *one* possible energy level for this particle, and therefore only one eigenstate ψ. It is equal to

$$\psi(x) = \frac{\sqrt{m\alpha}}{\hbar} e^{-\frac{m\alpha}{\hbar^2}|x|},$$

and E is conveniently $-\frac{m\alpha^2}{2\hbar^2}$, depending only on the mass of the particle m and the "depth" of the well α.

The hydrogen atom

In this case, our particle is an electron. Instead of measuring its potential V along an x-axis, we are now measuring it along r, the distance of the electron from the center of the atom. According to Coulomb, V is

$$\frac{-e^2}{4\pi\epsilon_0 r},$$

where e is *not* the famous irrational number, but the fundamental charge of the electron. Plugging this into Schrodinger's equation, we retrieve that

$$\psi_{nlm} = \sqrt{\left(\frac{2}{na_0}\right)^3 \frac{(n-l-1)!}{2n((n+1)!)^3}} e^{\frac{-r}{na_0}} \left(\frac{2r}{na_0}\right)^l L_{n-l-1}^{2l+1}\left(\frac{2r}{na_0}\right) Y_l^m(\theta,\phi),$$

where n, l, and m are all integers, a_0 is defined as $4\pi\epsilon_0\hbar^2/me^2$, and L and Y are mathematical "helper functions" that ease the calculation of ψ. More importantly, we have the slightly simpler equation

$$E_n = -\left(\frac{m}{2\hbar^2}\left(\frac{e^2}{4\pi\epsilon_0}\right)^2\right)\frac{1}{n^2}.$$

This means that starting from a specific energy E_1, the pitches we hear go up as $-1/n^2$. Easy!

Displaying the potential should be relatively simple — we can depict it as a line that falls off as $1/r$, since that's what it is. The difficult part comes when we want to display ψ itself on screen. What part of it should we show? The part that depends on r, θ, or ϕ? For now, let's just focus on r, and set θ and ϕ such that their part of the ψ equation comes out to a constant, and doesn't change with r. Therefore, we want to show an oscillating

$$\psi_{nl} = \sqrt{\left(\frac{2}{na_0}\right)^3 \frac{(n-l-1)!}{2n((n+1)!)^3}} e^{\frac{-r}{na_0}} \left(\frac{2r}{na_0}\right)^l L_{n-l-1}^{2l+1}\left(\frac{2r}{na_0}\right),$$

where $L_{q-p}^{p}(x)$ are the generalized Laguerre polynomials.

Chapter 3

Building the Program

3.1 Frameworks and Libraries

Our web application, tentatively titled “Listen to quantum mechanics,” is built solely on HTML, CSS, and JavaScript. The only external library in use is p5.js, a JavaScript framework based off Processing, which is in turn a “creative coding” framework based off Java. Since our program relies heavily on advanced visualization and auralization, p5.js comes into play when we want to display certain oscillating functions on screen, or generate a certain combination of waveforms. Its numerous features include drawing lines, shapes, and even single pixels to an HTML canvas, updating them at up to 60 frames per second, and, through its extension p5.sound.js, generating “oscillators” that can play any basic waveform (sine, triangle, square) at any audible pitch. We utilize all these features to do things that would be very difficult with standard HTML and CSS, such as draw the shape of each quantum potential, make every energy level user-interactable, and show the dynamic shape of the full wavefunction when multiple states are active.

3.2 Web Hosting

Since this program is built strictly for a web browser, it will run natively on any personal computer, regardless of its operating system. The program was initially built on Electron (electronjs.org), a framework used to build HTML applications that can be downloaded and run as standalone applications on a user's computer, but we decided to migrate this project fully to the web, since a) everyone has some sort of web browser, b) no one wants to download a 150-megabyte application when they can access the same thing easily through a simple URL. The main reason we started out with Electron was because p5 had compatibility issues with Safari, but since the beginning of this project, those have been resolved, and the application should now work on all major web browsers. The project is hosted at `https://itsbreese.com/quantum` and will remain there for the foreseeable future.

3.3 Index.js

`Index.js` iis the principal JavaScript file from which our entire program is run. It contains all the code for creating the HTML canvas; drawing the potential shapes, energy levels, and wavefunction eigenstates and superpositions; detecting user input and playing oscillators for the required eigenstates. Here is an outline of the file:

- `class UserData`
  - `constructor()`
  - `setView()`
  - `getEnergyLevel()`
  - `activateEnergyLevel()`
- `class Renderer`
  - `constructor()`
  - `drawPotential()`
  - `drawEnergyLevels()`
  - `drawWave()`
  - `drawHitboxes()`

```
│
├── class EnergyLevel
├── class QuantumMath
│   ├── static hermite()
│   ├── static factorial()
│   └── static getWavefunction()
└── setup(), draw(), mousePressed()
```

We have defined four classes that will help us organize our code for readability and efficiency: `UserData`, `Renderer`, `EnergyLevel`, and the "helper" class `QuantumMath`. The latter class contains two static functions, `hermite(n, x)` and `factorial(n)` that are self-explanatory: `hermite(n, x)` returns the n-th Hermite polynomial evaluated at x, and `factorial(n)` returns $n!$. It also contains `getWavefunction(p, n, x)` that returns the evaluated ψ_n at position x and a level n in a potential well p (with p being an integer from 0 to 3, representing our four options).

Returning to the top of the tree, the class `UserData` contains several key pieces of information including: which potential the user is looking at, how many energy states are "active" (i.e., how many oscillators are being played at once), the pitches of those oscillators, and the "base energy" for which the ground state of every potential is positioned at. It also contains the methods `setView()`, `getEnergyLevel()`, and `activateEnergyLevel()`, all of which are called in different cases when the user interacts with the screen with their mouse, which is first handled by the p5.js event `mousePressed()`, shown at the bottom of the tree. In a nutshell, `mousePressed()` sees where on the canvas the user has clicked, and then determines if an energy level should be "activated" by calling `getEnergyLevel()`. If an appropriate energy level is found, it is then activated with `activateEnergyLevel()`, which turns on an oscillator and increments `userData`'s number of active oscillators by one. The method `activateEnergyLevel()` also contains all the code for de-activating an energy level, so there's no need for a separate function to accomplish that.

While `userData` contains information about the user and what they should be seeing and hearing, the class `Renderer` contains information and methods explicitly for drawing to the screen. The methods `drawPotential()` and `drawEnergyLevels()` do what they imply, while `drawWave()` is used specifically for when one or more eigenstates are active — in place of drawing the energy level, this function draws the exact corresponding wavefunction to the screen in the same position. Like many other methods in this program, it is broken up into four cases for each of the four available potentials in this program. The ultimately unused method `drawHitboxes()` was at one point used for debugging — if called, it would show the "acceptable areas" in which the user could click that would activate certain energy levels.

The class `EnergyLevel` contains a height value (signifying where on the screen it should be drawn), a pitch value (noting what musical pitch it corresponds to), a Boolean flag noting whether it is active, and its own instance of a `p5.Oscillator` class — that is, the sound itself — that turns on and off when the user activates the energy level. The reason that `activateEnergyLevel()` is contained inside `userData` and not this class is because I preferred that all "mouse events" picked up by `mousePressed()` be directed to `userData`, rather than whatever class seemed most fitting, and since `userData` contains information for all energy levels on the screen, it was much easier to have that class interpret the event, rather than a single instance of an eigenstate.

Lastly, the three global functions `setup()`, `draw()`, and `mousePressed()` are all functions inherited from p5.js that place and size the canvas on the screen, refresh its contents, and detect mouse input, respectively.

3.4 Other Files

index.html

This HTML file is the "document outline" of the webpage. Like all HTML files, it has a title and styling, but the important part is that it calls index.js at the end.

It also contains some additional script for loading the sidebar on the page, namely the asynchronous function `fillSidebar()`. In a separate thread, this function gets information from `assets/potentials.json` through a JavaScript fetch request and fills it into the sidebar.

`index.html` also contains the drawing canvas on which all quantum-mechanics-related material is displayed, and, in the sidebar, the variable controls for changing parameters, and the list where the user can select the well they are viewing.

Chapter 4

Results and Conclusions

4.1 Conclusions

In one sentence: we've created a tool that allows a user to watch quantum-mechanical wavefunctions and hear the respective energies of one or more stationary states in four different simple potential wells.

This tool has a variety of applications, both for teachers and for students. It clearly depicts the relationship between a particle's attributes (mass, etc.), and its possible energy levels, while also presenting this quantum-mechanical behavior in a clean, intuitive manner. It could even be used to make quantum-inspired sound art.

Ultimately, this project turned out to be as much a learning experience in coding as it was an exploration of quantum mechanics. Not only did we make it easy to hear the relationships between various eigenstates, but I also came out of the experience with a much stronger footing on clean, organized, efficient code. In the next section, I'll detail some features that could still be fit into this program given some extra time, but after the near future, if we wanted to build a better program from the ground up, it might be a good idea to start from scratch — noting in more detail the goals we're setting out for ourselves. This way, we wouldn't run into the issue of having to recode large sections nearly every time we want to add something.

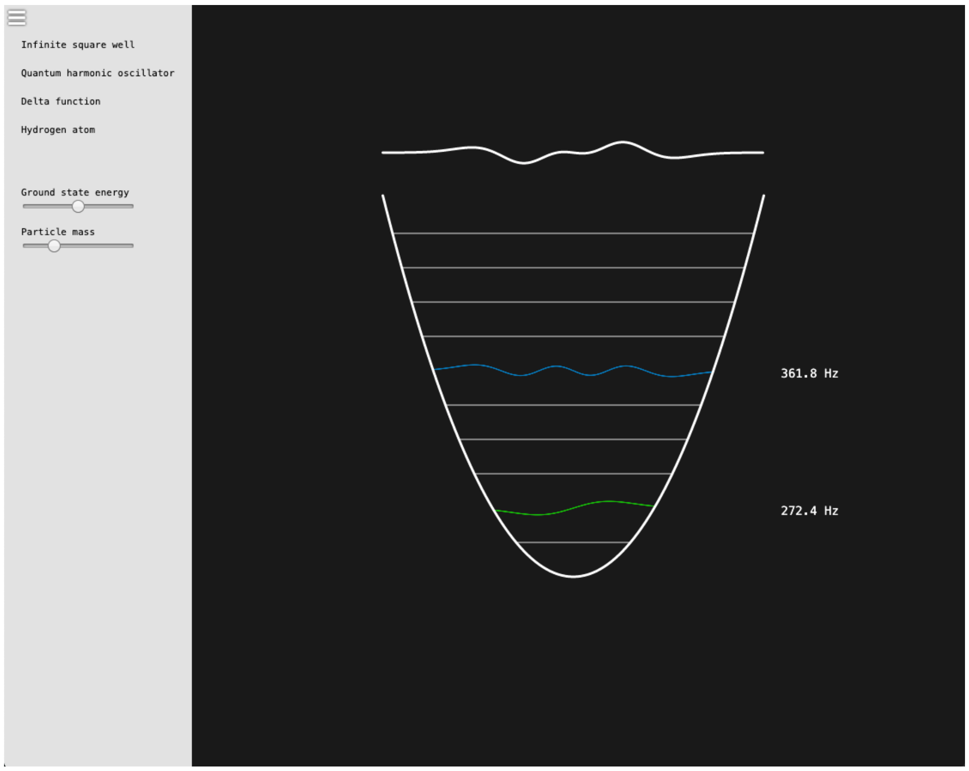

Figure 4.1: A screenshot of the completed app. The user has activated ψ_2 and ψ_6, resulting in a superposition, or a two-tone "chord" of 272.4 Hz and 361.8 Hz.

This project (or any future versions, should they supersede this current one) will be hosted at `https://itsbreese.com/quantum` for as long as I continue to run a website at that domain. Many thanks to Keith Griffioen, for giving me the inspiration for this project, as well as David Armstrong, for giving me a strong foundation in quantum mechanics.

4.2 Future Plans

There are several things we did not end up having time to add to this program, but could certainly be feasible given some additional effort. These include:

The step potential/finite square well

This is one of the few remaining quantum potentials that can be solved analytically, and yields predictable energy levels — it is quite similar to the results of the infinite square well, in fact. However, when a particle's energy exceeds that of the "step," the quantization breaks down a bit, and we start seeing different energies created by reflected and transmitted "wave packets." Ultimately, it proved beyond the scope of our capabilities given our timeframe. However, should work on this project continue, this potential would certainly be one of the priorities.

Modifiable proportions of eigenstates in superpositions

At the moment, if the user has more than one eigenstate ψ selected, the total wavefunction Ψ will be a sum of equal parts of all of them — that is, C_n is constant across all active frequencies. However, more code could be added into the program to present the user with additional sliders that appear on the sidebar when multiple eigenstates are selected, so they might be able to have 90% of ψ_1 and 10% of ψ_2, for example. This would not only change the relative volumes of the active energy levels — it would also drastically change the shape of Ψ.

Appendix A

Code samples

This is where we will post samples of the code we wrote for this project, to better illustrate the problems we were faced with and how we overcame them.

The following is the `EnergyLevel` class structure in full. Note the `hitbox` object, used to store the area in which the user can click to activate the energy level. Its `y` attribute stores where it is visually presented on the screen (this property is determined outside the class, in the `UserData` initialization), and its `pitch` attribute is determined in the method `assignPitch()`. The `switch` statement in this method splits it into four different functions, depending on which potential well the user is observing (`userData.potential`). It then grabs information from the user's controls in the sidebar, adjusting with "multipliers" to make the output pitches in an audible, non-painful range.

```
class EnergyLevel {

  constructor(y) {
    this.y = y
    this.hitbox = {lower:0, upper:0}
    this.active = false
    this.pitch = 0

    this.oscillator = new p5.Oscillator()
    this.oscillator.setType("sine")
    this.oscillator.freq(this.pitch)
    this.oscillator.amp(0.2)
  }
  assignPitch(level) {
    let baseEnergy = parseFloat(document.getElementById("baseEnergy").value)
    let multiplier = 0
    if (level != 2) {
      let mass = parseFloat(document.getElementById("mass").value)
    }
    switch (userData.potential) {
      case 0:
        let wellWidth = parseFloat(document.getElementById("wellWidth").value)/100
        multiplier = 1/(5*mass*sq(wellWidth))
        this.pitch = sq(multiplier*level) + baseEnergy - sq(multiplier)
        break
      case 1:
        multiplier = sqrt(1000/(parseFloat(document.getElementById("mass").value)/100))
        this.pitch = multiplier*(level - 0.5) -0.5*multiplier + baseEnergy
        break
      case 2:
        this.pitch = baseEnergy
        break
      case 3:
        this.pitch = -mass/sq(level) + baseEnergy + mass
        break
    }
    this.oscillator.freq(this.pitch)
  }

}
```

Next, we have a single method in the `Renderer` class, `Renderer.drawPotential()`. As mentioned previously, this method is responsible for drawing all four potential wells to the screen. Similarly to `assignPitch()` and many other methods in our program, it uses a `switch` statement to split into four cases based on the well.

```
drawPotential(potential) {
    stroke(255)
    strokeWeight(3)
    noFill()
    beginShape()
    switch(potential) {
      case 0:
      vertex(-this.size, this.size)
      vertex(-this.size, -this.size)
      vertex(this.size, -this.size)
      vertex(this.size, this.size)
      break
      case 1:
```

```
      for (var i = 0; i <= 100; i++) {
        let x = map(i, 0, 100, -this.size, this.size)
        vertex(x, 2*sq(x)/this.size-this.size)
      }
      break
      case 2:
      vertex(-this.size, this.size*0.8)
      vertex(-5, this.size*0.8)
      vertex(0, -this.size*0.8)
      vertex(5, this.size*0.8)
      vertex(this.size, this.size*0.8)
      break
      case 3:
      for (var i = 1; i < 100; i++) {
        let x = map(i*i, 1, 10000, -this.size, this.size)
        if (10/(x + this.size) - 1 < 1) {
          vertex(x, 10*this.size/(x + this.size) - this.size)
        }
      }
      break
    }
    endShape()
  }
```

The functions `vertex()`, `beginShape()`, and `endShape()` all come with p5.js. An interesting thing to note is our strategy for drawing the $1/r$-shaped potential of the hydrogen atom: if we drew one point in the shape at regular intervals, we would lose fidelity at the very beginning, creating an unpleasant jagged plot. However, by mapping each of our vertices' r value to their own square roots, we cram more of them at the beginning of the plot and fewer at the end, where the plot flattens out and less detail is needed.

Bibliography

[1] Alhosban, Fuad Hamad Mousa. “The Effectiveness of Aural Instructions with Visualisations in e-Learning Environments.” University of Durham (United Kingdom), 2011.

[2] Mayer, Richard E. Multimedia Learning 2nd ed. Cambridge, New York: Cambridge UP, 2009. Print.