Master thesis
Universität Bremen

Simulating a tilting maze using the Schrödinger Equation to illustrate quantum mechanical principles

Written by: Jonas Brenig
jbrenig@uni-bremen.de

Primary advisor:	Prof. Dr.-Ing. Udo Frese
Secondary advisor:	Dr. Philipp Niemann

August 27, 2021

Universität Bremen

Nachname Brenig Matrikelnr. 4237499

Vorname/n Jonas

Urheberrechtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.
Alle Stellen, die ich wörtlich oder sinngemäß aus anderen Werken entnommen habe, habe ich unter Angabe der Quellen als solche kenntlich gemacht.

Datum Unterschrift

Erklärung zur Veröffentlichung von Abschlussarbeiten

Ich bin damit einverstanden, dass meine Abschlussarbeit im Universitätsarchiv für wissenschaftliche Zwecke von Dritten eingesehen werden darf.

Datum Unterschrift

Contents

1 Introduction

Quantum mechanics and the Schrödinger equation are notorious for being difficult to understand and working against human intuition. Since quantum mechanics is a description of physics at the smallest scale possible, finding suitable visualizations is often difficult. Nonetheless, with the advent of quantum computers and the ongoing miniaturization of integrated circuits, the physics behind it is becoming more relevant than ever.

In an effort to lower the barrier of entry into learning quantum physics, in this thesis, I present a video game, which at its core simulates the equation governing the time evolution of quantum systems: the Schrödinger equation. There are a lot of projects using software and games to teach quantum mechanics. Ranging from virtual experiments to fully-fledged video games, these projects offer a more playful approach to quantum mechanics. As part of this thesis, "Schrödingers Labyrinth" was developed to provide players with a better understanding of the Schrödinger equation.

With the goal to shed more light on the time evolution of quantum mechanical systems, "Schrödingers Labyrinth" harnesses the power of modern GPUs to present time evolution of the wave function in real-time. Guided through multiple levels, the player can watch and control how the wave function behaves in a variety of situations by adjusting the potential field as if controlling a tilting labyrinth.

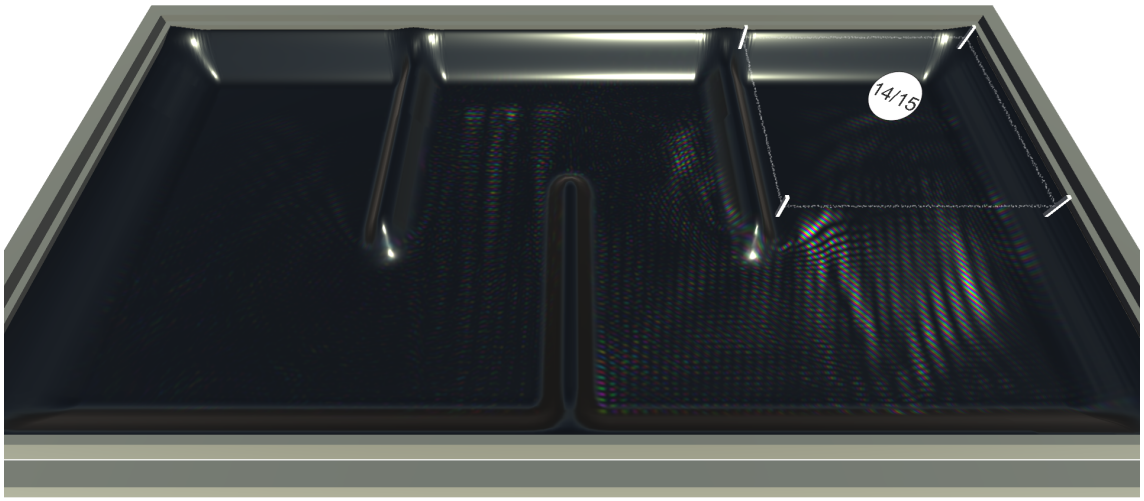


Figure 1.1: Ingame screenshot of Level 2

"Schrödingers Labyrinth" features multiple levels, with each level highlighting different aspects of quantum mechanics. At its core, the game is based on a numerical simulation of time evolution of a single particle in a 2-dimensional world. The split-step Fourier method is used to solve the Schrödinger equation. By implementing the algorithm to run in parallel on the GPU, a complex 2-dimensional world can be efficiently simulated in real-time.

While this cannot and is not intended to replace learning about quantum mechanics via other media, it can improve human intuition of how a wave function behaves in different environments. This thesis is divided into 4 major parts. First, in Chapter 2, we will take a look at the theoretical background of the Schrödinger equation and various phenomena of quantum mechanics. Chapter 3 deals with how the Schrödinger equation can be solved

numerically to provide a fast and reliable simulation that can be used in the game. Then, in Chapter 4, the actual game "Schrödingers Labyrinth" is presented, highlighting the core game mechanics, as well as commenting on some implementational details. Finally, Chapter 5 evaluates how the game was received and which areas have room for improvement.

2 Quantum mechanics

Before jumping into how the game works and how it was envisioned, some basic understanding of the underlying theory is necessary. Very small particles like electrons behave very differently than what we are used to from classical physics. Instead, the behavior of particles at a very small scale is described by quantum mechanics. At the heart of quantum mechanics, the state of any particle is expressed via its wave function Ψ.

When dealing with multiple particles, their wave function might depend on each other in some form. More formally a single wave function would describe both particles. However, dealing with multiple particles makes simulation significantly more expensive, as each particle adds a set of spatial parameters that need to be dealt with. Therefore we will only deal with a single particle.

> At multiple points of this chapter, you will find boxes like this one. These boxes aim to give a short, high-level overview of several aspects of quantum mechanics without diving deep into the math behind it.

In this chapter, I will give an overview of the quantum mechanical principles that are presented in the game as well as the necessary background needed to simulate the Schrödinger equation. This will by no means be a complete overview over quantum physics, rather focusing on basic intuition and theory needed to simulate time evolution. For a more in-depth look at quantum mechanics consider the lectures on quantum mechanics by Feynman [FLS11] or lectures provided by *MIT OpenCourseWare* [AEZ13].

Notation. *There are a variety of common notations in quantum mechanics. In this thesis, we will (mostly) follow the notation of the MIT Quantum Physics course 8.04 [AEZ13].*

2.1 The wave function

A particle in quantum mechanics does not have a definite position and speed. Instead, these properties are described via its wave function, which dictates which values for speed are possible and likely. This is one of the primary concepts of quantum mechanics and also the focus of this thesis, as the Schrödinger equation describes the evolution of the wave function over time.

> Instead of having definite values for properties such as position or speed, a particle's state in quantum mechanics is described completely by its wave function. The position of the particle depends on the amplitude of the wave function. Since the wave function is a complex function, it has a phase. This phase can be an indicator of the kinetic energy of the particle. A particle with a lot of differences in phase, like seen previously in Figure 1.1, generally has more energy.

The wave function of a single particle is a complex-valued function of time t and position $\vec{x}$ in n-dimensional space. In the context of this thesis, this usually means 2-dimensional space.

$$\Psi(\vec{x}, t) \in \mathbb{R}^n \times \mathbb{R} \to \mathbb{C} \tag{2.1}$$

It also needs to be *normalizable*:

$$\int \|\Psi(\vec{x}, t)\|^2 \, d^n\vec{x} = 1 \tag{2.2}$$

The function $\|\Psi(\vec{x}, t)\|^2$ is a probability density function describing the likelihood to find the particle at position $\vec{x}$ and time t.

Contrary to classical particles, particles in quantum mechanics (generally) do not have a specific position and velocity. Instead, they exist in a *superposition* of possible positions (and velocities). When measuring an observable (such as position or velocity) the probability of the measured value depends on the wave function Ψ.

Note that there are other formalisms to describe quantum mechanics, such as the density matrix, a more general description of a quantum state. These are often used when dealing with more complex systems or incomplete information [Neu27].

Since we are focusing on the simulation of the Schrödinger equation for simple, single-particle systems, we will only consider the wave function formalism.

2.1.1 Operators and eigenfunctions

An operator is a function acting on the wave function, transforming it in some way or another. The most simple operator is the identity operator $\hat{\mathcal{I}}$, which does not affect the wave function at all.

$$\hat{\mathcal{I}}\Psi = \Psi$$

Operators are used in quantum mechanics to describe the various properties of the wave function. Usually, we are particularly interested in the eigenfunctions and corresponding eigenvalues of an operator. The function Ψ is an eigenfunction of the operator $\hat{A}$ if and only if $\hat{A}$ acting on Ψ is equal to multiplication with a constant. This constant then is also called an eigenvalue.

$$\hat{A}\Psi = c \cdot \Psi$$

Each observable (eg. momentum, energy, or position) has its associated operator. If the particle is in an eigenstate of the given operator, meaning the particle's wave function is an eigenfunction of the given operator, the particle has a definite value for the given observable.

Since the state of a particle is completely described by its wave function, information about the particle's position and the particle's speed influence each other. For example, if the particle is in a state with definite speed, we have no information about the position of the particle. (See Figure 2.1)

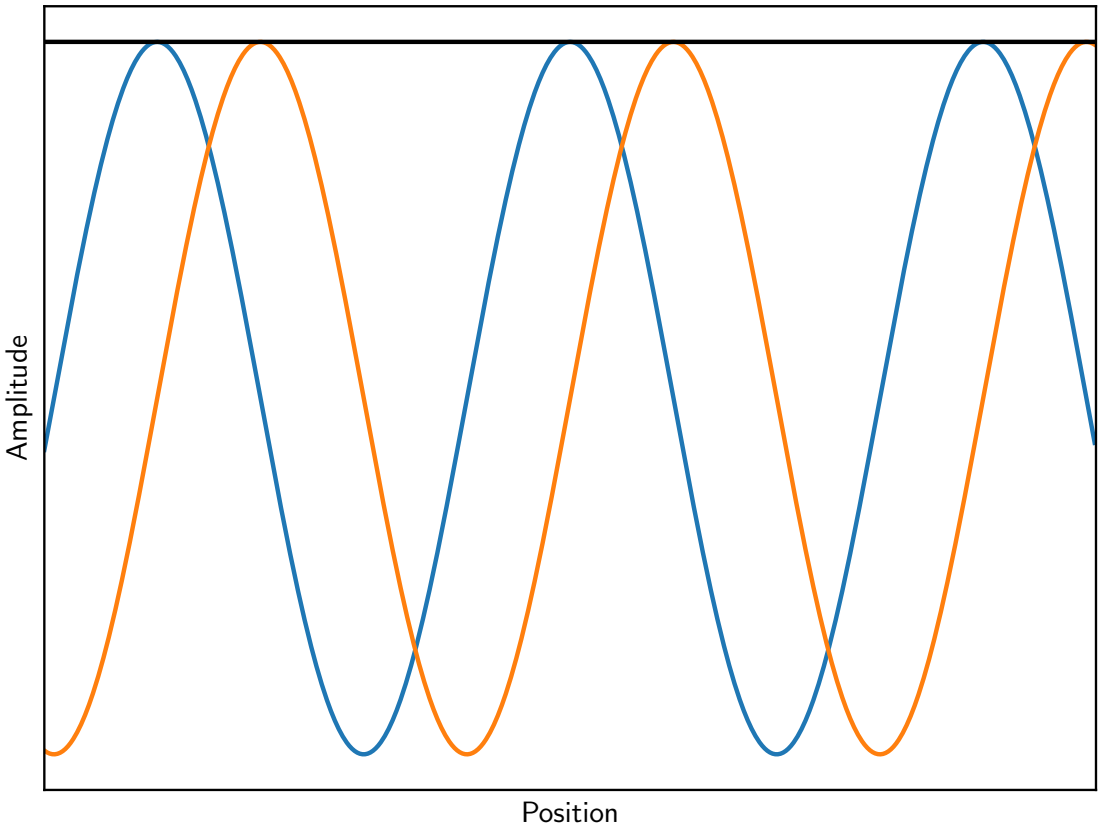


Figure 2.1: Plane wave in 1D. Real component in blue, imaginary in orange, probability density in black

For example, the eigenfunctions of the momentum operator $\hat{p}$ are plane waves. Figure 2.1 shows a 1-dimensional plane wave.

$$\hat{p} = -i\hbar\nabla \tag{2.3}$$

Here $\hbar = \frac{h}{2\pi}$ is the reduced *Planck's constant*. ∇ is a short-hand for gradient operator $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$.

$$\Psi(\vec{x}) = e^{i\vec{k}\vec{x}} \tag{2.4}$$

It is easy to see that plane waves, as constructed by Equation (2.4), are eigenfunctions of the momentum operator $\hat{p}$.

$$\hat{p}\Psi(\vec{x}) = -i\hbar\nabla e^{i\vec{k}\vec{x}} \tag{2.5}$$

$$= \hbar\vec{k} \cdot e^{i\vec{k}\vec{x}} \tag{2.6}$$

Here $\vec{k}$ is also called the *wavenumber* and is a measure of the particle's momentum. Since $\|e^{i\vec{k}\vec{x}}\|^2 = const$, knowing the exact momentum of a particle means knowing nothing about its position. Also note that this wave function is not realistic, because it is not normalizable (see Equation 2.2).

This also relates to the *Heisenberg uncertainty principle*, which states that $\Delta p \cdot \Delta x \geq \frac{\hbar}{2}$. When uncertainty in momentum Δp approaches zero, uncertainty in position Δx goes to infinity and vice versa.

Thanks to the Fourier theorem it immediately becomes apparent that any wave function Ψ can be expressed as a linear combination of eigenfunctions of momentum. The Fourier theorem states that any function Ψ can be expressed by the following integral:

$$\Psi(\vec{x}) = \frac{1}{\sqrt{2\pi}^n} \int_{\mathbb{R}^n} (\mathcal{F}\Psi)(\vec{k}) e^{i\vec{k}\vec{x}} d\vec{k} \tag{2.7}$$

As you can see the wave function Ψ is expressed entirely as an infinite linear combination of eigenfunctions of momentum, which are weighted by $(\mathcal{F}\Psi)(\vec{k})$, with $(\mathcal{F})$ being the Fourier transform.

As is the case here, the operators that are relevant to quantum physics usually form an eigenbasis, meaning using a linear combination of eigenfunctions of the given operator allows us to express any wave function possible. Apart from being used in the Schrödinger equation, operators and their eigenfunctions also play an important role regarding superposition.

2.1.2 Superposition

Superposition is what is usually meant when talking about a particle being in multiple states (usually referring, but not limited to multiple positions) at once. While this sounds like a superposition would be a very special and rare state, this is not the case. Looking at the wave function of a particle it becomes clear that a particle is actually always in a superposition of multiple states. Even if a particle has a definite measured position, its speed would be in a superposition and largely unknown.

> Quantum mechanics is known for particles that seem to be in multiple places at once. This phenomenon can be explained as a superposition of multiple wave functions, each describing the particle at a particular position.

A superposition is a linear combination of potentially infinitely many wave functions. Usually, we are interested in superpositions of multiple eigenfunctions of a particular operator. Since each observable has a corresponding set of eigenfunctions, expressing a wave function as a superposition of eigenfunctions of this operator can give us insights into the possible value of the observable.

Note that the complex constants C_i have to be chosen in a way that complies with the normalization condition of equation (2.2), as the squares of these constants $|C_i|^2$ give the probability density of the corresponding eigenfunction.

As a consequence, the probabilities of a measurement of an observable depend on these constants C_i. (We will take a closer at measurement look in Section 2.3)

$$\Psi(\vec{x}, t) = \sum\!\!\!\!\!\!\int_i C_i \Psi_i(\vec{x}, t) \tag{2.8}$$

The sum in equation (2.8) is superimposed with an integral symbol to represent the fact that some linear combinations may be uncountably infinite. This is the case, for example, when talking about the position of a particle. Obviously, when dealing with a numerical simulation later on, these will always be finite.

The wave function of a particle with a definite position is a Dirac-Delta function. Since the particle is at a specific position its probability to be at said position is 1. Therefore its wave function must integrate to 1 at the given position and 0 everywhere else. The Dirac-Delta function might be roughly characterized like this:

$$\delta(x) = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{2.9}$$

With the aforementioned condition to integrate to 1:

$$\int_{-\infty}^{\infty} \delta(x)\ dx = 1 \tag{2.10}$$

Extending this to multiple dimensions, any wave function Ψ can be expressed as a superposition of states of definite position, which are weighted by $C_{\vec{x_0}}$ as follows:

$$\Psi(\vec{x}) = \int_{\vec{x_0}} C_{\vec{x_0}} \delta(\vec{x} - \vec{x_o}) \tag{2.11}$$

As mentioned previously in Section 2.1.1, the eigenfunctions for momentum are plane waves. Since means we can decompose any wave function into eigenfunctions of the momentum operator using the Fourier transform, any wave function can be seen as a superposition of eigenfunctions of momentum. This will later be used in the split-step Fourier method in Section 3.4.

2.1.3 A wave packet

It is impossible to have no uncertainty in position and momentum at the same time, due to the *Heisenberg uncertainty principle*. However, a good compromise between uncertainty in position and momentum is a wave packet. See Figure 2.2 for an example of a 1-dimensional wave packet.

> A wave packet is a middle ground between uncertainty in position and momentum. Because a wave packet has uncertainty in momentum its uncertainty in position will increase over time. While the average position will move with constant speed, the size of the gaussian will increase.

A wave packet can be constructed by multiplying a plane wave with a Gaussian. We receive a wave function with only little uncertainty in both momentum and position. Equation (2.12) constructs a 1-dimensional wave packet.

$$\Psi(x) = \exp(ikx) \cdot \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big) \tag{2.12}$$

This construction can be easily extended to cover more than one dimension. As you can see position uncertainty is described by a gaussian, meaning that we know its average position μ and its uncertainty in form of the standard deviation σ.

Similarly, the wavenumber k is a measure for its mean momentum, with the uncertainty in momentum also being distributed like a gaussian. Figure 2.2 provides a 1-dimensional example of a wave packet. The real component is in blue, the imaginary component is in orange and the probability density function is in black.

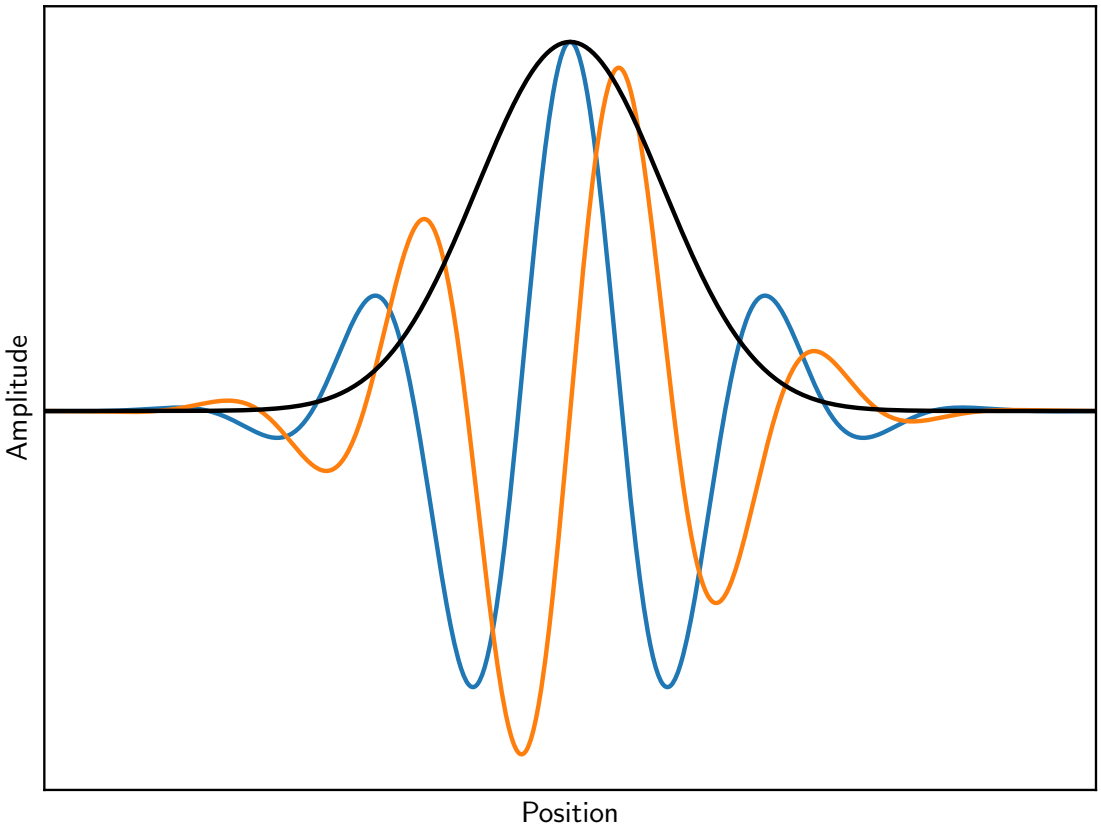


Figure 2.2: A 1-dimensional wave packet

2.2 The Schrödinger equation

The Schrödinger equation describes how the wave function (Ψ) of a quantum-mechanical system evolves in time. It will later be used as the core game mechanic in Chapter 4.

For a single particle the differential equation is as follows [FH65, Sch26]:

$$i\hbar\frac{\partial}{\partial t}\Psi(\vec{x},t) = \hat{H}\Psi(\vec{x},t) \tag{2.13}$$

Where t is time, x is the position, and $\hat{H}$ is the energy operator of the system.

The energy operator $\hat{H}$ for a single particle in a scalar potential consists of two components. The potential energy operator $\hat{V}$ describes the potential field, while the kinetic energy operator $\hat{T}$ describes the momentum of the particle.

$$\hat{H} = \hat{T} + \hat{V} \tag{2.14}$$

$$= \frac{\hat{p}^2}{2m} + \hat{V} \tag{2.15}$$

The momentum operator $\hat{p}$ in position space is defined as:

$$\hat{p} = -i\hbar\nabla \tag{2.16}$$

Where $\hbar = \frac{h}{2\pi}$ and h is *Planck's constant*. In Equation (2.16) we use the Nabla operator ∇ which is a short-hand for the gradient operator as introduced in Section 2.1.1: $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$. Therefore the energy operator in position space is:

$$\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{x}, t) \tag{2.17}$$

In Equation (2.17) the Nabla operator ∇ is squared. This squared Nabla operator is also called the Laplace operator and has the following definition for 2 dimensions:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

The complete Schrödinger Equation for a single particle in a scalar potential becomes:

$$\frac{\partial}{\partial t}\Psi(\vec{x}, t) = -\frac{i}{\hbar}\Big(-\frac{\hbar^2}{2m}\nabla^2\Psi(\vec{x}, t) + V(\vec{x}, t)\Psi(\vec{x}, t)\Big) \tag{2.18}$$

Example. *The most trivial system is the time evolution of a plane wave in a constant potential. In this case $\Psi = e^{i\vec{k}\vec{x}}$ is an eigenfunction of the energy operator (as well as the momentum operator).*

This means that $\hat{H}\Psi = E\Psi$, where $E = \frac{p^2}{2m} = \frac{\hbar^2|\vec{k}|^2}{2m}$. Solving the Schrödinger equation we get:

$$\frac{\partial}{\partial t}\Psi(\vec{x}, t) = -\frac{i}{\hbar}E \cdot \Psi(\vec{x}, t) \tag{2.19}$$

$$= -\frac{i\hbar|\vec{k}|^2}{2m} \cdot \Psi(\vec{x}, t) \tag{2.20}$$

$$\Rightarrow \Psi(\vec{x}, t) = \exp(-\frac{i\hbar|\vec{k}|^2}{2m} \cdot t) \cdot \Psi(\vec{x}) \tag{2.21}$$

Therefore Ψ evolves over time only by rotating some phase ω: $\Psi = e^{-i\omega t}e^{i\vec{k}\vec{x}}$

States like the one in the example are often referred to as stationary states. Any wave function that is an eigenfunction of the systems energy operator $\hat{H}$ is in such a stationary state. When these states evolve over time they only rotate their phase, which means that their probability distribution $|\Psi|^2$ does not change over time. States like this obviously rely on a constant potential.

Since the Schrödinger equation is a linear differential equation, a decomposition of Ψ into eigenstates of the energy operator $\hat{E}$ can be used as a general solution of the Schrödinger equation.

$$\Psi(\vec{x}, t) = \sum_n C_n e^{-iE_n t}\Psi_{E_n}(x) \tag{2.22}$$

However, finding the eigenfunctions for an arbitrary potential is not feasible. Therefore other (numerical) methods are needed to solve the Schrödinger equation in general. See Chapter 3 for more details.

2.2.1 Interference

A very important consequence of the Schrödinger equation is that the wave function can interfere. This causes a lot of classically unintuitive behavior. A very well-known example of this is the double-slit experiment, which demonstrates how particles can exhibit both the behavior of a wave and the behavior of a classical particle (when measured).

Interference is a consequence of how wave functions get combined in superpositions. When adding two wave functions together their probability density function might exhibit interference effects. That means in general the sum of probability densities of two wave

functions $|\Psi_1|^2 + |\Psi_2|^2$ might not be the same as the probability density $|\Psi_1 + \Psi_2|^2$ of the sum of these two wave functions. Obviously, there might be situations without interference where the probability density function remains the same.

$$|\Psi_1|^2 + |\Psi_2|^2 \stackrel{?}{\neq} |\Psi_1 + \Psi_2|^2 \tag{2.23}$$

In Figure 2.4, the particle is in a superposition of two wave packets moving towards each other (see Figure 2.3). Instead of simply adding the probability densities of the two wave packets (remember, the probability density of a wave packet is a gaussian), a superposition combines the complex wave functions. As a consequence interference effects can be observed in the probability density function of the resulting superposition. The probability density function is in black, the components of the complex wave functions are in blue (real) and orange (imaginary). At some positions the wave function interferes constructively, increasing the probability of the particle to be found at said position, while at other positions the wave function interferes destructively, decreasing the probability.

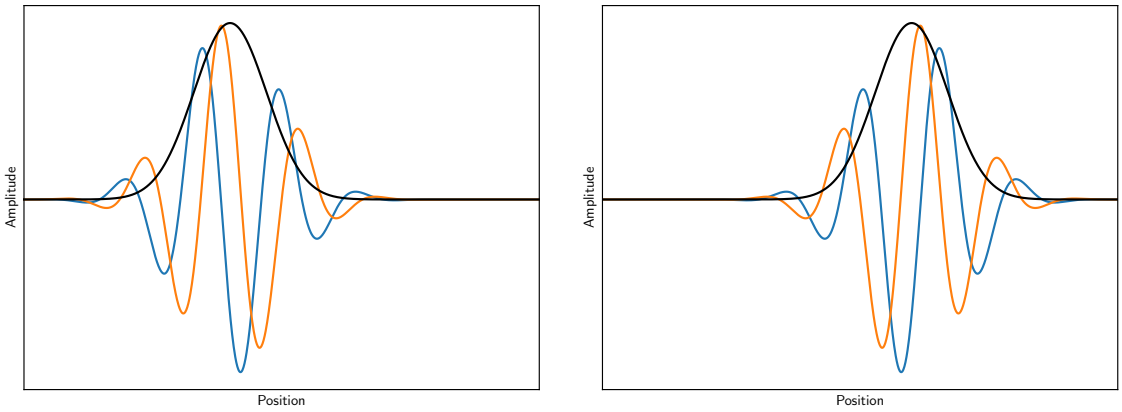


Figure 2.3: Wavefunctions before interference

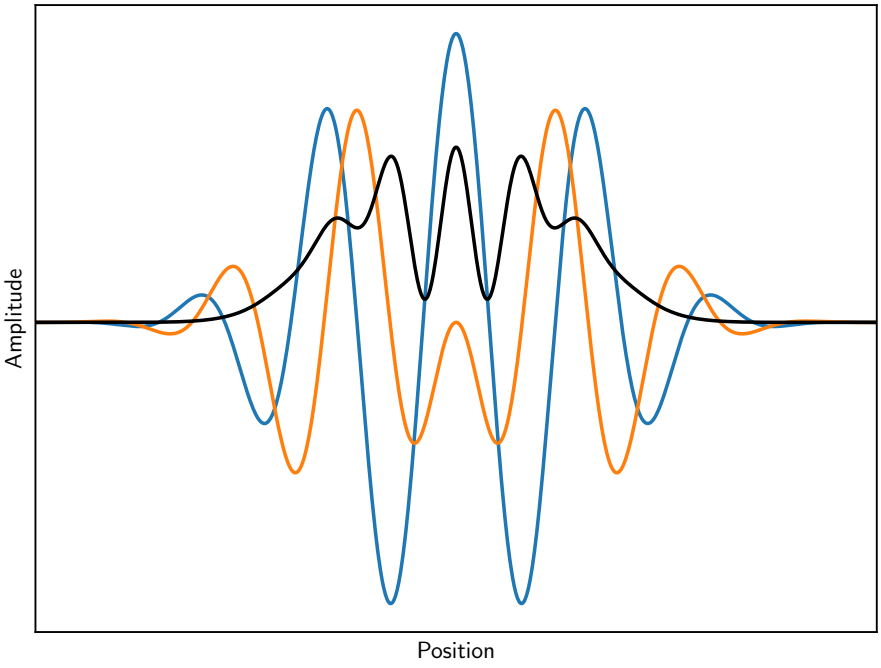


Figure 2.4: Example of interference effects

2.2.2 Quantum tunneling

When encountering a potential barrier (a part of the potential with energy higher than the energy of the particle) a classical particle will never *tunnel through*. Since the particle does not have enough energy to climb the potential, it will instead *roll* back down. The wave function of a particle in quantum physics however will not immediately have zero amplitude inside the classically disallowed region. Instead, the amplitude will decay exponentially, meaning the particle will have $> 0\%$ probability to be found inside the barrier. You can see this exponential decay inside the classically disallowed area in Figure 2.5.

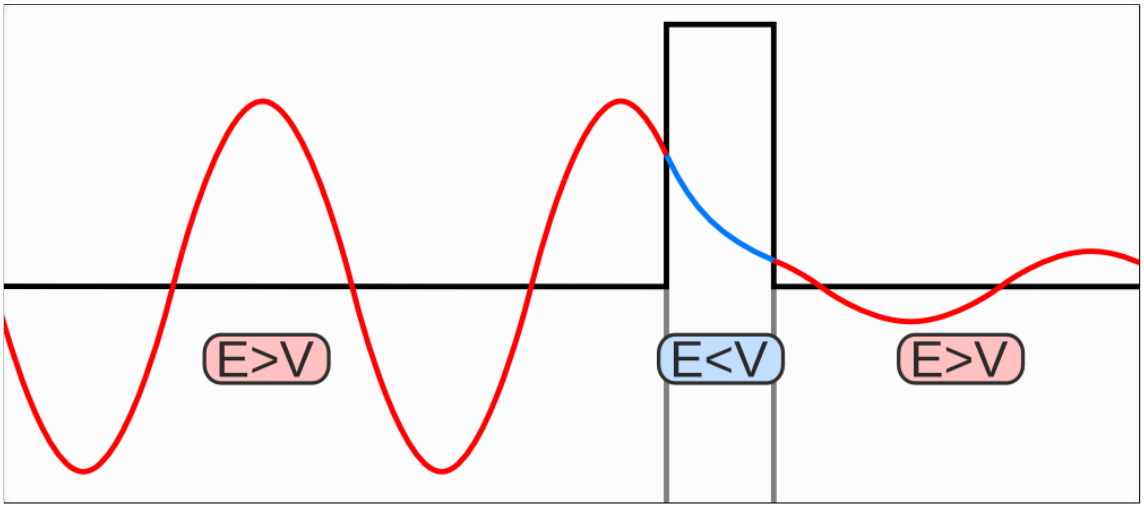


Figure 2.5: Quantum Tunneling. Graphic by Felix Kling[1]

This means if the barrier is small enough, part of the wave function might continue after the barrier with lower amplitude. In this case, there is a small probability to find the particle behind the barrier; the particle *tunneled* through. The amount of the wave function that will tunnel through depends on the energy difference between the particle and the barrier, as well as the thickness of the barrier. Obviously, a particle with higher energy than the potential barrier will be able to pass through anyway.

2.2.3 Resonance in the finite potential well

A classical particle will always continue through when encountering a potential well as seen in Figure 2.6. The potential energy difference will get converted into kinetic energy when entering the well and be converted *back* when exiting. Afterward, the particle will continue at the same speed as before. Of course, we are always talking about a nice world without friction.

> Resonance in a finite potential well is an example of very unintuitive behavior of quantum mechanics. In the classical world, a particle will never reflect when encountering a drop in potential energy. The wave function however can reflect, possibly even multiple times. Note that the wave function will never get trapped inside since no friction is modeled.

[1]Graphic by Felix Kling: `https://en.wikipedia.org/wiki/File:TunnelEffektKling1.png`

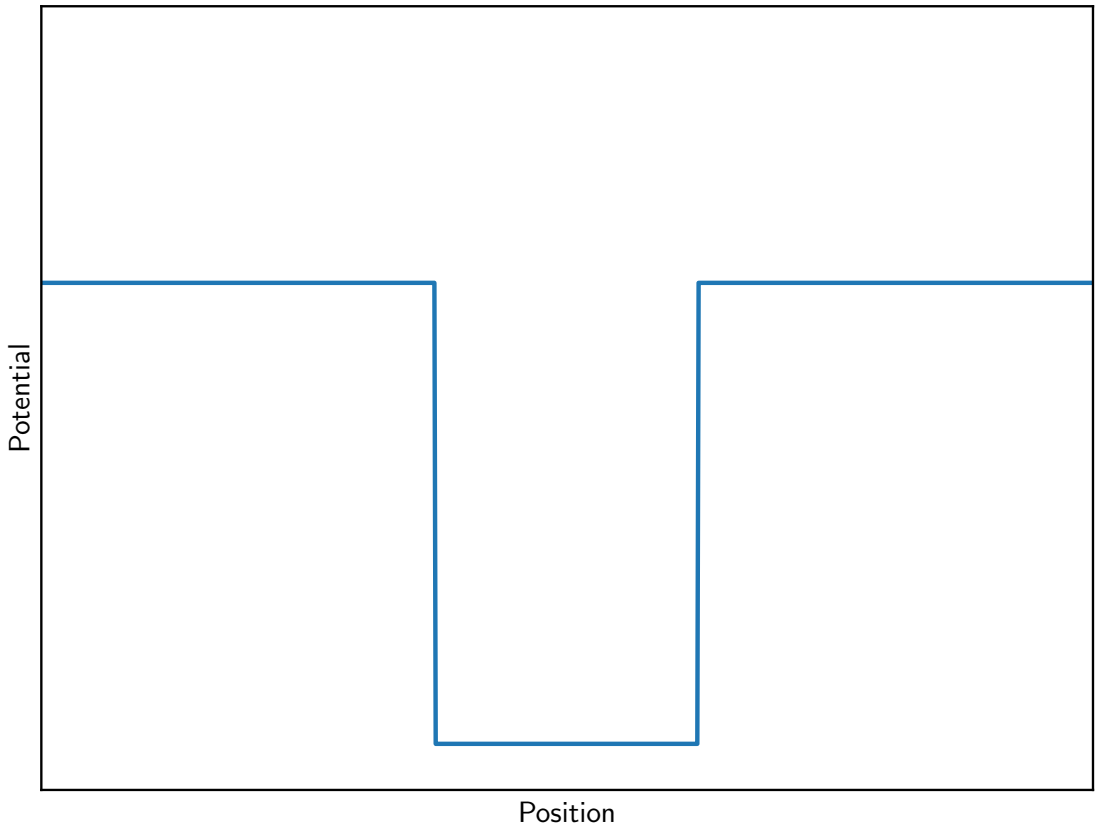


Figure 2.6: Illustration of a potential well

The wave packet of a particle in quantum mechanics however might also reflect off the potential well, depending on the internal energy of the particle. If the wave packet has enough energy it might also transmit through the potential well, just like a classical particle would. More accurately, the higher the kinetic energy of the wave packet is, the higher the amount of the wave packet that transmits through the well will be.

Interestingly at special wavelengths, which depend on the length of the well, the amount of the wave packet that transmits through the potential well increases significantly. This allows a wave packet to transmit almost completely even though a wave packet that is just slightly slower or slightly faster might reflect [MPS10].

In particular, the length of the potential well needs to be a multiple of $k\pi$, with k being the wavenumber of the wave packet. (See Section 2.1.3)

2.3 Measurement

Measurement is the second way of how the wave function can evolve over time. Usually, we take for granted that any particle behaves just the way we measured it. But because a measurement will always introduce *classical components* to the quantum mechanical system, it will behave very differently. Whenever any observable of a particle such as position or velocity is measured the wave function collapses into a possible state.

> A measurement is a significant break in the deterministic time evolution of the Schrödinger equation. Whenever a measurement occurs the wave function collapses into a state that represents the measured outcome.

Any arbitrary wave function will collapse into an eigenfunction after measurement. So after measuring the particle at position x_0 the wave function will be $\Psi = \delta(x - x_0)$.

The probability of each possible eigenvalue depends on the wave function itself. As previously discussed in Section 2.1.2, the wave function can be expressed as a linear combination of eigenfunctions of the given operator. If the wave function is of form $\Psi = \sum_i C_i \Psi_i$, the value $|C_i|^2$ gives the probability density of measuring the corresponding eigenvalue.

Therefore, when talking about positions $\Psi(\vec{x}) = \int_{\vec{x_0}} C_{\vec{x_0}} \delta(\vec{x} - \vec{x_o})$ represents the wave function. The probability of measuring the particle to be at position $\vec{x_0}$ then is $|C_{\vec{x_0}}|^2$.

Since the act of measuring the particle always requires an interaction of the particle with its *classical environment*, the change of the wave function is not covered by the Schrödinger equation anymore. Usually, this is described using the concept of quantum decoherence [Zeh70, Sch05]. After a measurement was done, the wave function will continue to evolve according to the Schrödinger equation.

3 Simulation

The Schrödinger equation can be solved analytically for some potentials. However, when simulating arbitrary potentials a numerical solution method is needed. In this chapter, we will look into which methods are available and how the simulation was implemented.

There are a variety of methods to numerically solve the Schrödinger Equations for arbitrary systems. In order to use such a simulation in a real-time environment such as a game, the calculation of a single simulation step has to be done very efficiently. For the purposes of this thesis, I have targeted a maximum execution time of 33ms, which amounts to roughly 30 simulation steps per second.

Another factor is the stability of the chosen method. While the simulation does not need to be incredibly accurate, it needs to be accurate and stable enough to produce plausible results. In order to do so, the norm $\int |\Psi(x)|^2 dx = 1$ of the wave function needs to be invariant under the chosen time evolution algorithm.

While the presented game uses a GPU-based implementation of the Split-step Fourier algorithm (see Section 3.4), some CPU-based alternatives were explored. Some of them were only usable in a 1-dimensional prototype, while others were efficient enough to work in 2-dimensions.

3.1 Discretization of time and space

In order to numerically simulate the time evolution of a wave function, time and space need to be discretized. Time is advanced by set intervals of Δt. Space is divided into a uniform grid. The distance between two neighboring grid cells is Δx. Since changing Δt and Δx directly impacts the accuracy of the simulation, they need to be minimized while also achieving sufficient performance.

Note that special handling of the ends of the grid is needed. The easiest approach is to assume a circular world in which indices wrap around. So in a grid with dimensions of $X \times Y$ the wave function wraps around at its borders: $\Psi(0,0) = \Psi(X,Y)$.

Other boundary conditions are also thinkable but might be harder to implement, while not providing noticeable benefits for gameplay. Since the split-step Fourier method, due to its reliance on the Fourier transform, implicitly assumes a circular world we opted for the easiest approach: Modeling a circular world and limiting the world artificially by creating potential barriers at its ends.

It would be possible to impose more complex boundary conditions, but this will most likely not provide any notable benefits to the gameplay experience. Such an approach might be more appropriate for simulations that require very high simulation accuracy [AES+03].

Notation. *In the following chapters $\Psi_t(x)$ will refer to the discretized wave function $\Psi_t(x) \in \mathbb{N}^n \to \mathbb{C}$ at time t with spatial index x. Similarly $V_t(x)$ will refer to the discretized potential. The time parameter of the wave function will usually be written in subscript. Especially when the spatial parameter is omitted.*

$$\Psi(\vec{x}, t) \hat{=} \Psi_t(\vec{x})$$

3.2 Naive Forward Euler method

The most simple way to do numerical time evolution calculates the current derivative and applies it to the current state.

$$\Psi_{t+\Delta t} = \Psi_t + \Delta t \cdot \Psi'_t \tag{3.1}$$

This approach leads to a relatively simple solution with the following derivative for Ψ:

$$\Psi'_t(\vec{x}) = -\frac{i}{\hbar}\Big(-\frac{\hbar^2}{2m}\nabla^2\Psi_t(\vec{x}) + V_t(\vec{x})\Psi_t(\vec{x})\Big) \tag{3.2}$$

In order to calculate Ψ'_t, the Laplacian is needed. The Laplacian at any grid cell can be estimated by looking at the neighboring grid cells. For two dimensions the Laplacian is estimated as follows:

$$\begin{aligned}\nabla^2\Psi(x,y) \approx \frac{-2\Psi(x,y) + \Psi(x-\Delta x, y) + \Psi(x+\Delta x, y)}{\Delta x^2} \\ + \frac{-2\Psi(x,y) + \Psi(x, y-\Delta y) + \Psi(x, y+\Delta y)}{\Delta y^2}\end{aligned} \tag{3.3}$$

After calculating $\nabla^2\Psi(x,y)$ only some multiplication and addition is necessary to estimate $\Psi_{t+\Delta t}$. (See Schrödinger Equation (3.2))

While this method is very easy to compute, it does not preserve the norm of the wave function as required by Equation (2.2). This introduces significant inaccuracies over time making this method unsuitable for any long-term simulation of time evolution.

3.3 Crank-Nicholson scheme

The Crank-Nicholson scheme combines the Forward Euler approach with the Backward Euler [FSS+09, CN47]. The Backward Euler uses the derivative at time $t+\Delta t$ to calculate $\Psi_{t+\Delta t}$.

$$\Psi_{t+\Delta t} = \Psi_t + \Delta t \cdot \Psi'_{t+\Delta t} \tag{3.4}$$

By combining both approaches, we get a time-evolution scheme that is known to conserve the probability distribution condition of Equation (2.2) [Vis91]. This means after a time-evolution step using this method, the overall probability mass will stay the same, delivering a valid wave function.

$$\Psi_{t+\Delta t} = \Psi_t + \frac{1}{2}\Delta t \cdot \Psi'_t + \frac{1}{2}\Delta t \cdot \Psi'_{t+\Delta t} \tag{3.5}$$

$$= \Psi_t - \frac{i}{\hbar}\hat{H}\Delta t\frac{1}{2}(\Psi_t + \Psi_{t+\Delta t}) \tag{3.6}$$

Since the derivative $\Psi'_{t+\Delta t}$ depends on the solution, it is necessary to solve a system of linear equations in order to calculate $\Psi_{t+\Delta t}$.

Assuming for now that $\hbar = 1$ and $m = 1$ results in the following equation:

$$\Psi_{t+\Delta t} = \Psi_t + \frac{1}{2}i\Delta t \cdot \Big(\frac{1}{2}\nabla^2\Psi_t - V\Psi_t\Big) + \frac{1}{2}i\Delta t \cdot \Big(\frac{1}{2}\nabla^2\Psi_{t+\Delta t} - V\Psi_{t+\Delta t}\Big) \tag{3.7}$$

$$-\frac{4i}{\Delta t}\Psi_{t+\Delta t} = -\frac{4i}{\Delta t}\Psi_t + \nabla^2\Psi_t - 2V\Psi_t + \nabla^2\Psi_{t+\Delta t} - 2V\Psi_{t+\Delta t} \tag{3.8}$$

The needed Laplacian is estimated the same way as described in Section 3.2. The following Equation (3.9) only uses a single dimension. It is trivial to extend to multiple dimensions.

$$\begin{aligned} -\frac{4i}{\Delta t}\Psi_{t+\Delta t}(x) = -\frac{4i}{\Delta t}\Psi_t(x) - 2V(x)\cdot\Psi_t(x) - 2V(x)\cdot\Psi_{t+\Delta t}(x) \\ + \frac{-2\Psi_t(x) + \Psi_t(x-\Delta x) + \Psi_t(x+\Delta x)}{\Delta x^2} \\ + \frac{-2\Psi_{t+\Delta t}(x) + \Psi_{t+\Delta t}(x-\Delta x) + \Psi_{t+\Delta t}(x+\Delta x)}{\Delta x^2} \end{aligned} \tag{3.9}$$

After rearranging, the following equation can be solved using a system of linear equations.

$$\begin{aligned} &\frac{4i}{\Delta t}\Psi_t(x) + 2V(x)\cdot\Psi_t(x) + \frac{2\Psi_t(x) - \Psi_t(x-\Delta x) - \Psi_t(x+\Delta x)}{\Delta x^2} \\ &= \frac{4i}{\Delta t}\Psi_{t+\Delta t}(x) - 2V(x)\cdot\Psi_{t+\Delta t}(x) + \frac{-2\Psi_{t+\Delta t}(x) + \Psi_{t+\Delta t}(x-\Delta x) + \Psi_{t+\Delta t}(x+\Delta x)}{\Delta x^2} \end{aligned} \tag{3.10}$$

For 1-dimensional systems, this results in a tridiagonal matrix which can be solved in $O(n)$ time [Dat10].

$$\begin{bmatrix} b & c & 0 & 0 & 0 \\ a & b & c & 0 & 0 \\ 0 & a & b & c & 0 \\ 0 & 0 & a & b & c \\ 0 & 0 & 0 & a & b \end{bmatrix} \tag{3.11}$$

A tridiagonal matrix is a matrix where only the three central diagonals have values other than 0, as seen in Equation (3.11).

3.3.1 Implementing Crank-Nicholson on the CPU

As previously discussed, using the Crank-Nicholson scheme to simulate quantum time evolution is a very nice approach with good accuracy. When implementing this scheme for 1-dimensional systems it also is very efficient, as solving tridiagonal matrices can be done in $\mathcal{O}(n)$-time [Dat10].

However, when moving to 2-dimensional (or even higher) systems this is generally not the case anymore. It is still possible to use sparse matrix solvers, such as Eigen[1], that utilize things like Cholesky decompositions, but this approach is still significantly slower in comparison.

Both, a 1-dimensional prototype, as well as a 2-dimensional prototype were implemented using `C#` as well as necessary `C++` bindings. While this simulation scheme seems to provide sufficient accuracy, the performance problems for 2-dimensional systems meant that only very small systems could be simulated with appropriate speed. As a consequence, this would severely limit the spatial resolution of the game.

3.4 Split-step Fourier method

The split-step Fourier method splits time evolution into two phases. The Fourier transform is used to apply the kinetic energy operator in momentum space and the potential operator in position space [DOST96].

[1] Eigen homepage: `https://eigen.tuxfamily.org/index.php?title=Main_Page`

Assuming a constant energy operator $\hat{H}$, time evolution according to the Schrödinger equation has the following solution:

$$\Psi_{t+\Delta t}(\vec{x}) = e^{-\frac{i}{\hbar}\Delta t\hat{H}}\Psi_t(\vec{x}) \tag{3.12}$$

Since the momentum operator in position space contains a partial derivative, this does not work. For this reason, the energy operator $\hat{H}$ is split into $\hat{T}$ and $\hat{V}$ which are applied after each other. The potential energy operator $\hat{V}$ can now easily be applied since it only requires multiplication with $V(\vec{x})$.

$$\Psi_{t+\Delta t}(\vec{x}) = e^{-\frac{i}{\hbar}\Delta t\hat{H}}\Psi_t(\vec{x}) \tag{3.13}$$

$$\approx e^{-\frac{i}{\hbar}\Delta t\hat{T}}e^{-\frac{i}{\hbar}\Delta t\hat{V}}\Psi_t(\vec{x}) \tag{3.14}$$

The introduced error can be slightly reduced by applying half a timestep of the potential operator before and half a timestep after applying the kinetic energy operator. This is also known as Strang Splitting [Str68].

$$\Psi_{t+\Delta t}(\vec{x}) \approx e^{-\frac{1}{2}\frac{i}{\hbar}\Delta t\hat{V}}e^{-\frac{i}{\hbar}\Delta t\hat{T}}e^{-\frac{1}{2}\frac{i}{\hbar}\Delta t\hat{V}}\Psi_t(\vec{x}) \tag{3.15}$$

While calculating the effects of the potential operator in position space is a simple multiplication, the effects of the kinetic energy operator cannot be calculated in the same manner. Instead, the Fourier transform $\mathcal{F}$ is used to apply the kinetic energy operator $\hat{T}$ in momentum space. Instead of needing to calculate the Laplacian, the kinetic energy part of time evolution becomes a simple multiplication too.

$$\Psi_{t+\Delta t}(x) \approx e^{-\frac{1}{2}\frac{i}{\hbar}\Delta t\hat{V}}\mathcal{F}^{-1}\big(e^{-\frac{i}{\hbar}\Delta t\hat{T}}\mathcal{F}(e^{-\frac{1}{2}\frac{i}{\hbar}\Delta t\hat{V}}\Psi_t)\big) \tag{3.16}$$

The complete algorithm has 5 steps. First, the potential energy part of the Schrödinger equation gets applied. After transforming the result into momentum space, the kinetic energy part is applied in momentum space. Then the wave function is transformed back into position space and the second half of the potential energy part gets applied.

$$\phi_1(\vec{x}) = e^{-\frac{1}{2}\frac{i}{\hbar}\Delta tV_t(\vec{x})}\Psi_t(\vec{x}) \tag{3.17}$$

$$\phi_2(\vec{k}) = \mathcal{F}\phi_1 \tag{3.18}$$

$$\phi_3(\vec{k}) = e^{-i\hbar\Delta t\frac{\|\vec{k}\|^2}{2m}}\phi_2(\vec{k}) \tag{3.19}$$

$$\phi_4(\vec{x}) = \mathcal{F}^{-1}\phi_3 \tag{3.20}$$

$$\Psi_{t+\Delta t}(\vec{x}) \approx e^{-\frac{1}{2}\frac{i}{\hbar}\Delta tV_t(\vec{x})}\phi_4(\vec{x}) \tag{3.21}$$

Besides the Fourier transforms, only multiplication of the wave function with a complex factor is needed. When using an efficient implementation of the Fourier transform, this means the algorithm is very fast, especially compared to the Crank-Nicholson scheme that was discussed in the previous section. Thankfully there are many implementations of the *Fast Fourier Transform* available. Furthermore, this also means that the algorithm can be efficiently implemented on the GPU.

3.4.1 Implementing Split-step Fourier on the CPU

Implementing the Split-step Fourier algorithm on the CPU requires a fast implementation of the Fourier transform. Luckily the *Intel Math Kernel Library*[WZS+14] provides a fast, hardware-accelerated implementation which was used to implement the CPU-based prototype.

Since no solving of linear equations is required, the Split-step Fourier method is significantly faster to compute than the Crank-Nicholson method. In particular, it is performant enough to allow simulations of larger 2-dimensional systems.

Since the prototype was implemented in `C#` it could easily be ported to the Unity game engine. Next up I extended this implementation to use GPU compute shaders to further increase simulation performance.

3.4.2 Implementing Split-step Fourier using GPU Compute

In order to further accelerate the simulation, the Split-step Fourier method was implemented on the GPU. This allows for significantly larger simulation grids, while still ensuring a smooth gameplay experience. All shaders used are written in the Unity compute shader language which itself is based on *HLSL*.[2]

The resulting implementation vastly outperforms the CPU-based alternatives, allowing for much better spatial resolution.

Fourier transform

The most computationally heavy parts of the algorithm are the necessary Fourier transforms. A very commonly used, GPU-based implementation of the Fast Fourier Transform (FFT) is provided by NVIDIA and called *cuFFT*.[3]

However, in order to stay platform-independent a solution implemented using *Unity Compute Shaders* was selected instead. For this purpose a slightly modified open-source implementation[4] of the Cooley-Tukey FFT Algorithm was used [LBG08].

A slight disadvantage of the selected implementation is, that it limits the grid size to powers of two. However, since we are using the Unity terrain feature to visualize the potential (see Section 4.6.3), we need to adhere to this limit anyway.

Schrödinger evolution

Applying the actual operators requires only multiplication with each grid cell which can be easily parallelized on the GPU.

Since Unity compute shaders do not natively support operations on complex numbers these had to be implemented manually. This includes implementation of the exponential function needed for the *Schrödinger equation*. In particular exponential functions of the form e^{ix} are needed, as seen in Section 3.4.

$$e^{ix} = cos(x) + i\ sin(x) \tag{3.22}$$

Implementations of the cosine and sine functions on the GPU often are only accurate for values ranging from $-\pi$ to π. For this reason, values passed to these functions are adjusted using modulo 2π in order to keep close to this value range.

The actual calculation for half of the potential energy part of the split-step Fourier method becomes the following. Remember that this step gets executed two times (see Section 3.4).

[2] High-level shader language (HLSL): `https://docs.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl`

[3] NVIDIA cuFFT: `https://developer.nvidia.com/cufft`

[4] FFT implementation: `https://github.com/nobnak/FftUnity`

$$\phi_1(x,y) = e^{-\frac{i}{2\hbar}\Delta t \cdot V(x,y)}\Psi_t(x,y) \tag{3.23}$$

The kinetic energy part of the Schrödinger equation requires the wavenumber, which is equivalent to the index of the Fourier transform. Some scaling and shifting are required, to account for negative indices and spatial scaling of the grid.

When the kinetic energy step is calculated, first the rescaling factors s_x and s_y for k are computed. These factors remain constant as long as the scaling parameters of the simulation stay the same. The variables w and h refer to the width and height of the simulation respectively. The mostly constant factor f is also precomputed on the CPU.

$$f = -\frac{\Delta t\hbar}{2m} \tag{3.24}$$

$$s_x = \frac{2\pi}{\Delta x \cdot w} \tag{3.25}$$

$$s_y = \frac{2\pi}{\Delta y \cdot h} \tag{3.26}$$

On the GPU only the following calculations remain:

$$\phi_3(x,y) = exp\Big(if(s_x^2 \cdot k_x^2 + s_y^2 \cdot k_y^2)\Big)\phi_2(x,y) \tag{3.27}$$

With k being the corrected index of the Fourier transform ranging from $\begin{pmatrix}-w/2\\-h/2\end{pmatrix}$ to $\begin{pmatrix}w/2-1\\h/2-1\end{pmatrix}$.

Parameters of the simulation

In principle, it is possible to run the simulation without using accurate real-world constants. Setting mass and $\hbar$ to 1 simplifies the math slightly, but requires special tuning to adjust the simulation scale to be sensible. In order to simplify this, parameters were chosen to represent real values. The selected particle mass is the electron. Timestep and grid-scale were chosen accordingly. With one grid cell being $10^{-10}m$ and one simulation step simulating $1.2 \cdot 10^{-16}s$ (at 60 frames per second). The potential field is also scaled accordingly, ranging from 0 to $2eV$.

3.5 Other methods

While the split-step Fourier and the Crank-Nicholson scheme are very popular options to solve the Schrödinger equation numerically, there are other methods to do find numerical solutions to the Schrödinger Equation. The paper "Numerical approaches to time evolution of complex quantum systems" gives an overview about some approaches [FSS+09].

One of the most promising alternative approaches is the *Chebyshev scheme*. While it is quite complex, it seems to provide good accuracy and efficient calculation. However we did not pursue this approach, as the split-step Fourier method allowed easy calculation on the GPU, allowing for sufficient performance and accuracy [Bae00, NS99].

4 The game: "Schrödingers Labyrinth"

Designing a game that integrates quantum mechanics brings with it its own difficulties. It is necessary to weigh against each other the accuracy and closeness to physics and the approachability of the game. In this chapter, we will take a look at how the presented game is designed and implemented. For this purpose, we will roughly follow the formal and dramatic elements of a game as outlined in the *Game Design Workshop* by Tracy Fullerton [Ful19].

The implemented game "Schrödingers Labyrinth" follows a unique design approach by closely integrating the Schrödinger equation as its core game mechanic. While this means the game is very close to the actual physics behind the simulation, it also limits the freedom in level design to a certain degree. Nonetheless, quantum mechanics still allows for enough variety in behavior to provide a good challenge to the player.

A gameplay demo of the game is available at `https://youtu.be/U7A9l2PEmK8`.

4.1 Related works

With quantum mechanics becoming important in many areas, there are a lot of software solutions that try to teach concepts of quantum mechanics. From simulators for quantum computing to virtual experiments to games that incorporate elements of quantum physics into their game design, a lot of approaches have been tried.

The variety of software available ranges from simulations of quantum computing, over virtual experiments designed as complementary teaching material, to more playful games that are based on quantum mechanical concepts.

Quvis

Some of the more learning-oriented software solutions include *Quvis*[1] of the *University of St Andrews*, which presents a variety of experiments including background information and is targeted to be included in lectures about the topic.

Since the simulations are specifically adjusted to present these experiments, they can cover a wide variety of topics. Even more complex topics like quantum cryptography are available. As you can see in Figure 4.1, the provided simulations are very restrictive in which parameters the user can control. This allows *Quvis* to provide a wide variety of experiments, which are accompanied by additional explanations and catered to students of quantum mechanics.

[1] Quvis: `https://www.st-andrews.ac.uk/physics/quvis/`

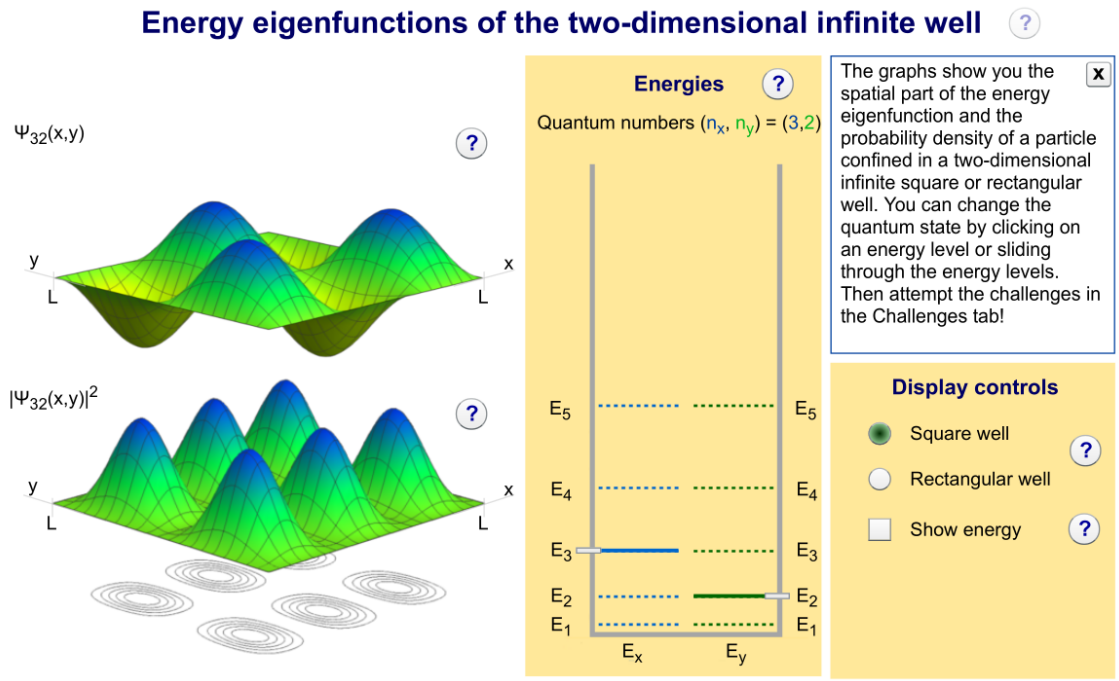


Figure 4.1: Quvis: 2-dimensional potential experiment

Quantum Game

Moving more into the direction of actual games *Quantum Game*[2] recreates a virtual laboratory, demonstrating various effects of quantum mechanics. The game is currently only available as an alpha version of the level editor. This virtual lab can be used to recreate various experiments of quantum mechanics. Furthermore, going into the direction of quantum computing, *Quantum Game* also provides elements dealing with qubits.

As you can see in Figure 4.2, similar to *Quvis*, *Quantum Game* also does not deal directly with the wave function but focuses more on probabilities and numerical properties of the particle.

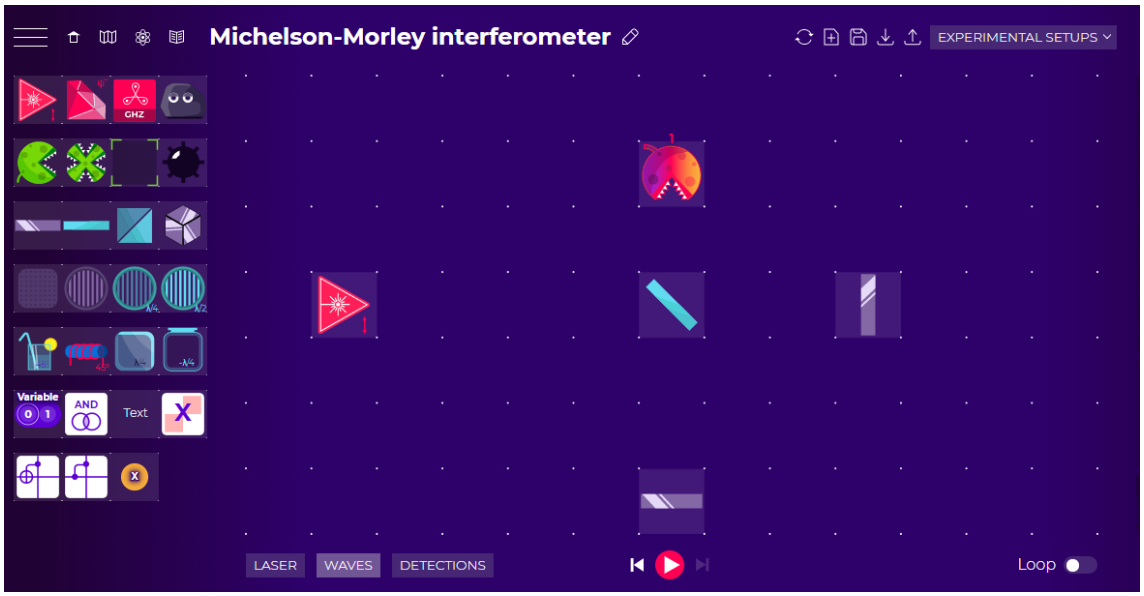


Figure 4.2: Quantum Game: Michelson-Morley Interferometer

[2]Quantum Game: `https://quantumgame.io/`

Particle in a Box & Psi and Delta

Particle in a Box, as well as *Psi and Delta* are two small games provided by a Team of the *Design and Social Justice Studio* of *Georgia Tech*[3]. These platformer games aim to teach some concepts of quantum mechanics by demonstrating them via their game design. Both games teach about energy eigenfunctions and probabilities, as well as some other related concepts of quantum mechanics.

Particle in a Box (Figure 4.3) focuses on the intuitive differences between the classical and the quantum mechanical world. Psi & Delta (Figure 4.4) features 15 different levels and also allows a second player to join.

Figure 4.3: Particle in a Box: Demonstrating energy levels

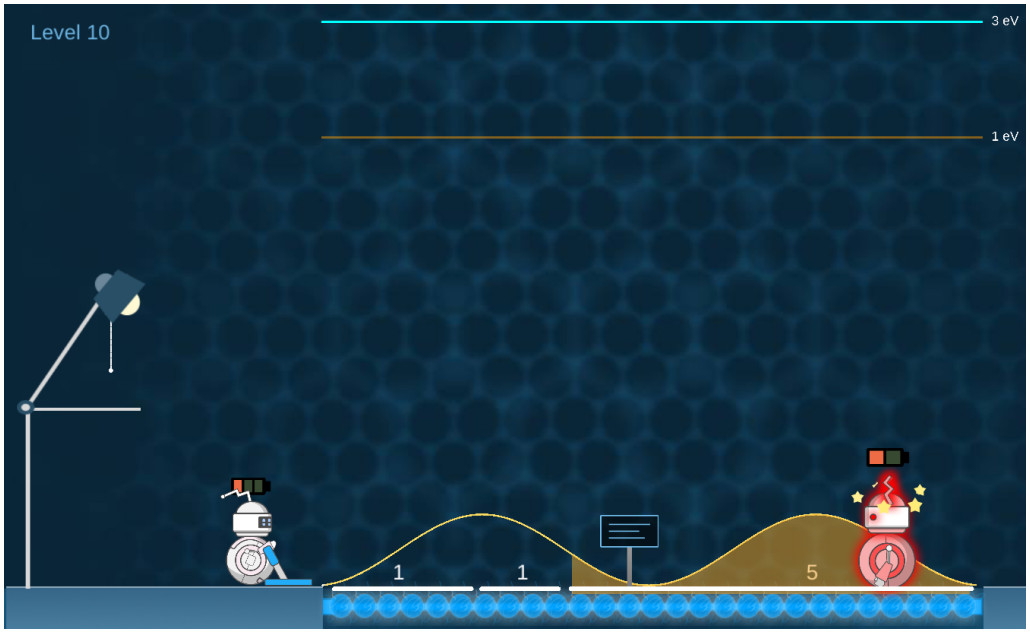


Figure 4.4: Psi & Delta: Demonstrating probabilities

[3]Learn Quantum Mechanics: `https://learnqm.gatech.edu/`

Quantum Moves 2

Quantum Moves 2[4] is a game provided by the *Science at Home* project. Not only does it keep very close to the physics, actually directly working with a 1-dimensional wave function, it also is used to generate data to contribute to ongoing research (similar to other games of the *Science at Home* project).

The target of the game is to move the wave function in a very specific way by controlling the potential. The player then is awarded a score, depending on how well the wave function represents the target shape. Additionally, the help of an optimizer can be employed, to further improve the recorded solution.

Quantum Moves 2 is similar to the game presented in this thesis, in that it also deals with time evolution of the wave function, albeit only using 1-dimensional wave functions.

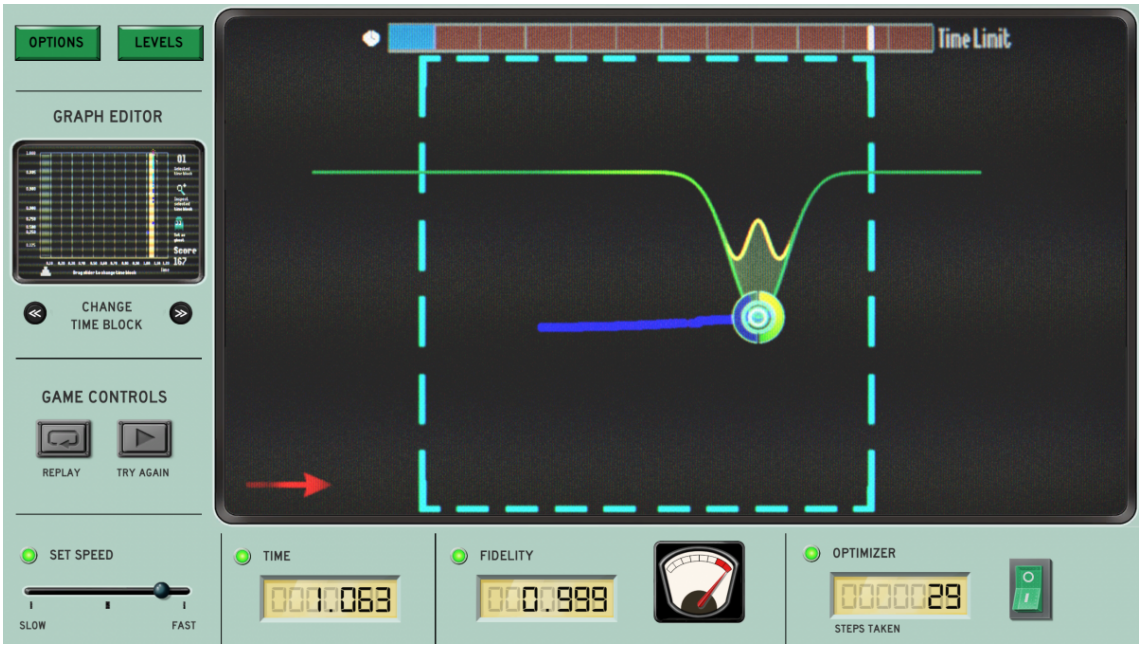


Figure 4.5: Quantum Moves 2

TestTubeGames

While not directly related to Quantum Physics, *TestTubeGames*[5] provides a handful of physics-related games teaching about several different areas of physics. I specifically mention these games, because they are structured in a similar manner as "Schrödingers Labyrinth", usually intermixing level-based gameplay with additional information presented via text boxes.

In Figure 4.6 you can see a screenshot of the game *Velocity Raptor*, which teaches about relativity. Featured as the main game mechanics are length contraction and doppler-shift.

[4]Quantum Moves 2: `https://www.scienceathome.org/games/quantum-moves-2/`
[5]TestTubeGames: `https://testtubegames.com/`

Figure 4.6: Velocity Raptor by TestTubeGames

4.2 Premise, Story and Character

The premise of the presented game, "Schrödingers Labyrinth", is built around an unnamed cat character, who was miniaturized to a quantum level and is trapped inside a tilting labyrinth. This is, of course, a reference to the well-known thought experiment of Schrödinger's cat.

Figure 4.7: Excerpt of the introductory level dialog

The player is tasked with helping the cat escape from imprisonment inside the labyrinth. However, being miniaturized it cannot act itself, instead it interacts with the player only through dialog. Because the cat was miniaturized to the level of quantum mechanics, it does not have a definite position. Instead, like every other particle of quantum mechanics (see Chapter 2), its state is defined by its wave function.

Each level, as the levels get harder and harder, the cat comes closer to the final exit. Finally, after the player has completed the last level, the cat can escape.

The accompanying dialog introduces the player to each level and comments on how well they are doing. For some of the more difficult levels additional hints are provided, that get shown if the player had to restart the level multiple times.

4.3 Challenge and Play

Player controls are limited to tilting the labyrinth, which directly influences the potential. Since the wave function will react to changes in the potential, this allows the player to steer the particle through the level, similar to how one would steer the marble in a classical tilting labyrinth. Similar to the classical tilting labyrinth each level is surrounded by walls. The player is provided with a direct visual representation of the 2-dimensional wave function that evolves according to the Schrödinger equation, as outlined in Chapters 2 and 3.

Of course, a wave function behaves very differently from a classical marble. Advancing through the levels requires intuition about how the wave function (representing the cat) reacts to certain changes in the potential. In particular, the wave function might spread out over time, reflect off walls and even interfere with itself. The ability to finish the level and score points depends on how localized the particle is. Therefore an indirect time limit is imposed on the player, since spreading the wave function might lead to a situation, where it is not possible anymore to beat the current level. As such, uncertainty in position is the key resource that has to be managed by the player. We will come back to how scoring works in detail in Section 4.7.

In contrast to a classical labyrinth, it is theoretically possible to gain enough energy to climb over the levels walls. Since the world is modeled circular (see Section 3.1), this allows the wave function to wrap around, circumventing most of the level architecture. However, as the particle has a lot of uncertainty in momentum at that point, it will not be possible to concentrate enough probability density in one place to beat the level. This automatically enforces the player to keep within the boundaries of the labyrinth.

In order to beat a level, some intuitive understanding of time evolution in quantum mechanics is necessary. While the first few levels are quite easy, later on the difficulty increases. Later levels have increasingly difficult obstacles, requiring a better understanding of how a wave packet reacts to certain situations.

The levels themselves are quite linear in the sense that they usually have a very clear path to the finish. The challenge lies in finding the right approach to navigate the various obstacles. Largely this means anticipating how the wave function reacts to different speeds and how it reflects off the levels walls. Usually, this means that there are not a lot of completely different solutions to beat a given level. However, for some levels trying different approaches might be necessary to increase the player's score.

4.4 Aesthetics

The wave function is visualized by encoding its phase in color hue and its square magnitude in brightness. This creates a very readable representation of the wave function with sufficient contrast to make out areas where the wave function has the highest amplitude.

Remark. *Unlike Figure 4.8, most screenshots in this thesis use a slightly modified version of the game, which has a white background instead of the dark default background. This was done to increase image quality when printed on paper.*

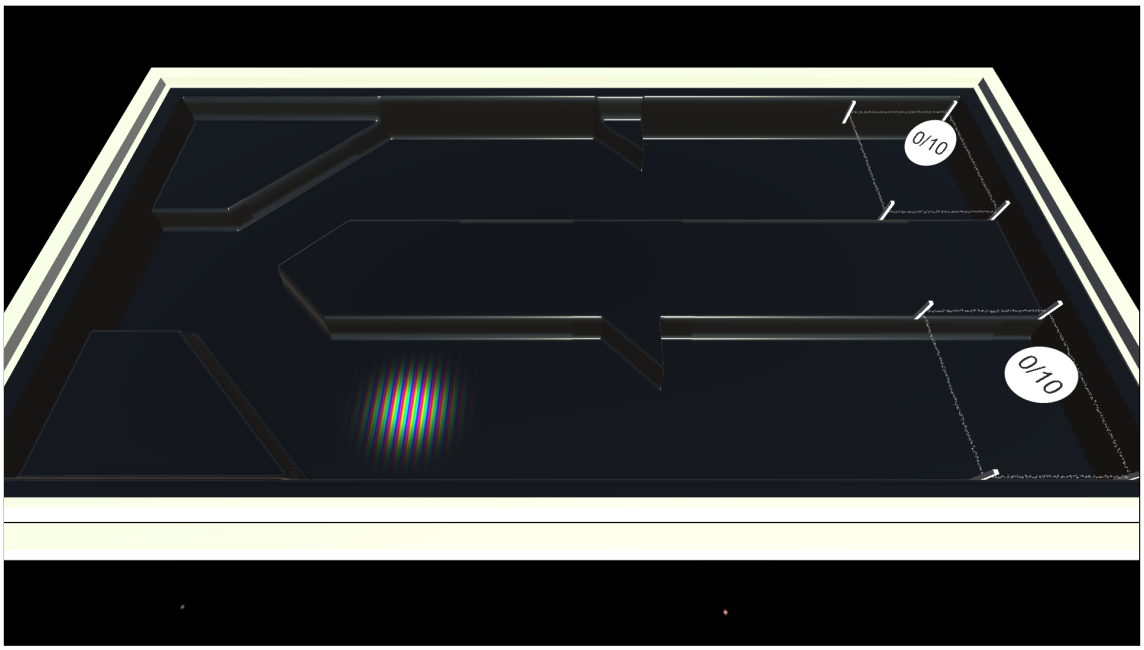


Figure 4.8: Visual representation of a wave packet

In Figure 4.8, you can see the visual representation of a 2-dimensional wave packet as it is presented in-game. This wave packet is a 2-dimensional version of the wave packet introduced in Section 2.1.3, with momentum along the x-axis. As you can see, the brightness encodes the position of the particle. Color is used to communicate the complex phase of the wave function. In this case this can be used to identify, that the wave function has momentum along the x-axis.

Since the wave function is defined over the complete space, most of this visualization will usually be very dark. This motivates the overall look and feel of the game, which is predominately black, with only the wave function providing color. In order to keep in theme, additional level geometry is held very simplistic, doing without complex textures that distract from the wave function.

The wave function representation with its rainbow-like aesthetic defines the core aspect of the game's visuals. This rainbow aesthetic is also used as an accent color in several important elements of the user interface. Similarly, other level elements are mostly shades of gray, only sparingly using color. The skybox background imitates a night sky, with stars subtly shifting in color, reinforcing the overall theme of phase.

4.4.1 Implementation of the visualization of the wave function

The wave function is internally represented using a two-channel texture (since it contains complex numbers). For rendering the wave function this representation gets converted into HSV color space, using phase for hue and squared magnitude for the color's value. Saturation remains fixed at 1 but could be altered for example to not show the phase of the wave function.

Since the wave function might contain very small values, especially when the particle's position has high uncertainty, the magnitude is scaled such that the brightest spot of the

texture always has value 1. For this purpose, a shader-based upon a prefix sum shader[6] calculates the maximum magnitude of the wave function.

4.4.2 Visualizing the potential using the third dimension

Since we are only simulating a 2-dimensional particle, we can use the third dimension to visualize the potential. This creates the visuals of a labyrinth as intended. Walls, slopes, or ditches in the potential can create a complex level architecture, allowing easily readable levels. This approach also further supports the aesthetic design of the game.

Representation of the potential in darkness

Since the visual representation of the wave function is very dark for most of the space, this often leads to situations where height differences of the potential were very hard to identify. Three solutions were explored to make the potential more readable for the player:

- Overlaying a grid texture similar to a wireframe
- Using edge detection shaders to create outlines around walls
- Using lighting and physically based rendering to create a natural shine around changes in height.

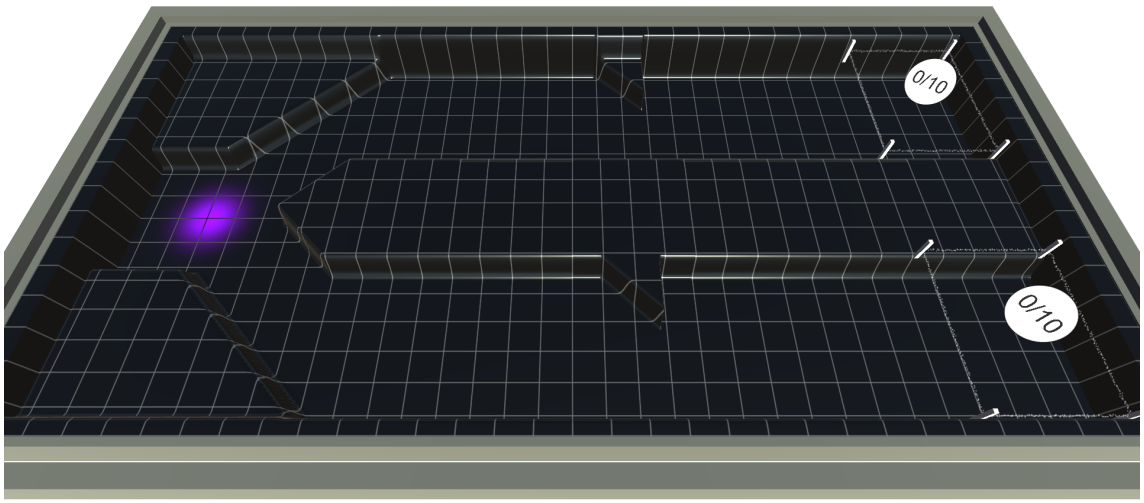


Figure 4.9: Level 2 with enabled grid lines

Overlaying a grid texture, as seen in Figure 4.9, is certainly a very simple and effective solution. It allows the player to easily determine how the potential is shaped. However, there are also some artifacts around areas with very steep differences in potential height, as the texture was stretched by the unity terrain engine. Also, this solution hurts the rather clean aesthetic of the rest of the game.

While using edge detection to create an outline around walls seemed very promising, given the wide variety of potentials no reliable solution was found and all approaches were highly dependent on the camera angle. Also, this approach does not deal with soft slopes, limiting its usefulness.

The biggest problem when using lights and their reflections to communicate height differences is that very sharp corners often do not reflect the light even though they would in reality. In the end, this was mitigated by using multiple light sources for some levels and

[6]PrefixSum shader: `https://github.com/walbourn/directx-sdk-samples`

applying slight Gaussian smoothing to the heightmap, rounding its corners, and therefore allowing more light to hit. The Gaussian Blur has to be applied on the GPU in order to maintain performance. Once again we opted for an open-source solution already available for Unity[7].

4.5 Level structure

The seven levels increase in difficulty, with each level introducing new concepts and providing more challenging obstacles. While the beginning is very tame and does not require a good understanding of how the wave function evolves over time, the later levels introduce some more complicated effects of quantum physics.

4.5.1 Intro and levels 1 & 2

In these levels, the player has some time to familiarize themselves with the controls and the way a wave function reacts to changes of the potential. While no specific concept is featured prominently, a lot of very basic understanding can be gained by playing these rather simple levels.

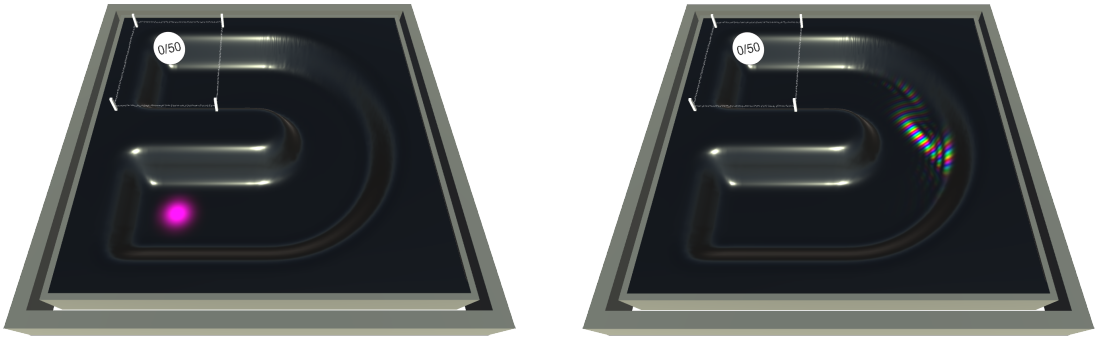


Figure 4.10: Intro Level: Learning the controls

Of these three levels, feedback showed that level 1 is the most difficult. It is also the only level to require some thought to solve. As you can see in Figure 4.11 the player starts in a hole. In order to escape it is necessary to build up some momentum by moving the particle back and forth within this hole. This level is also a good example of the complex potentials that can be realized.

Level 2, which is shown in Figure 4.12, is very easy conceptually. The difficulty of this level lies in the mechanical skill that is required to steer the wave function through the slalom-like labyrinth.

Together, these three levels are designed to let players familiarize themselves with the controls and the wave function. The specific phenomena of quantum mechanics that were introduced in Chapter 2 are demonstrated in the levels following afterward.

[7]Gaussian Blur shader: `https://github.com/keijiro/GaussianBlur`

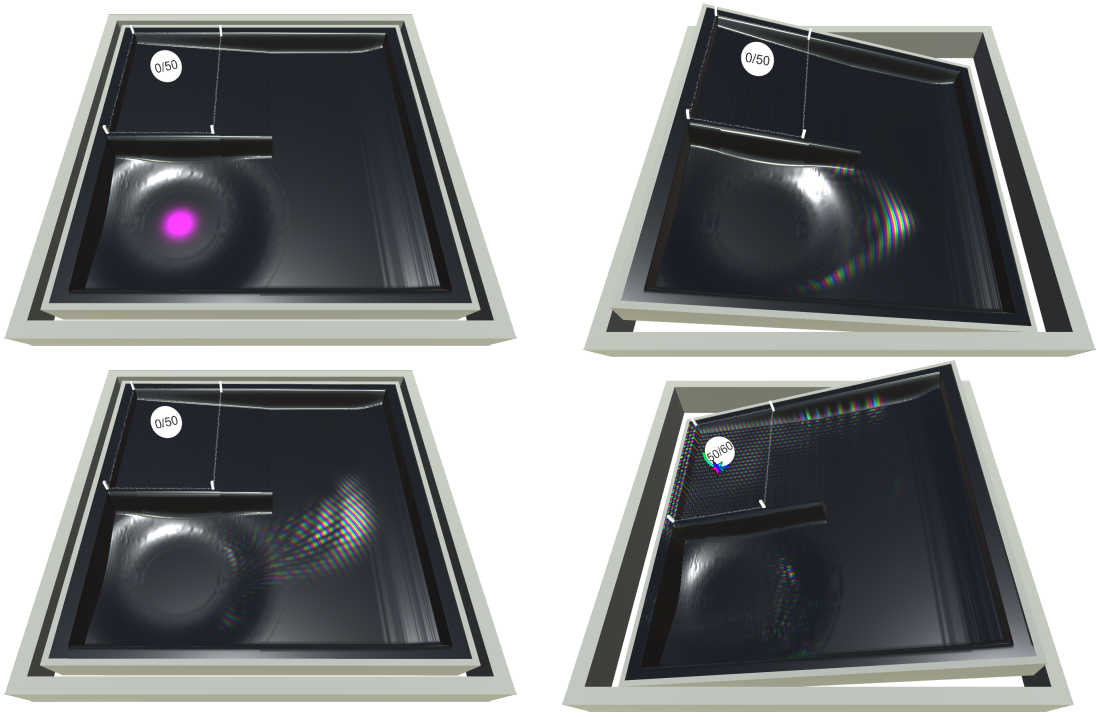


Figure 4.11: Level 1: Building up momentum to escape

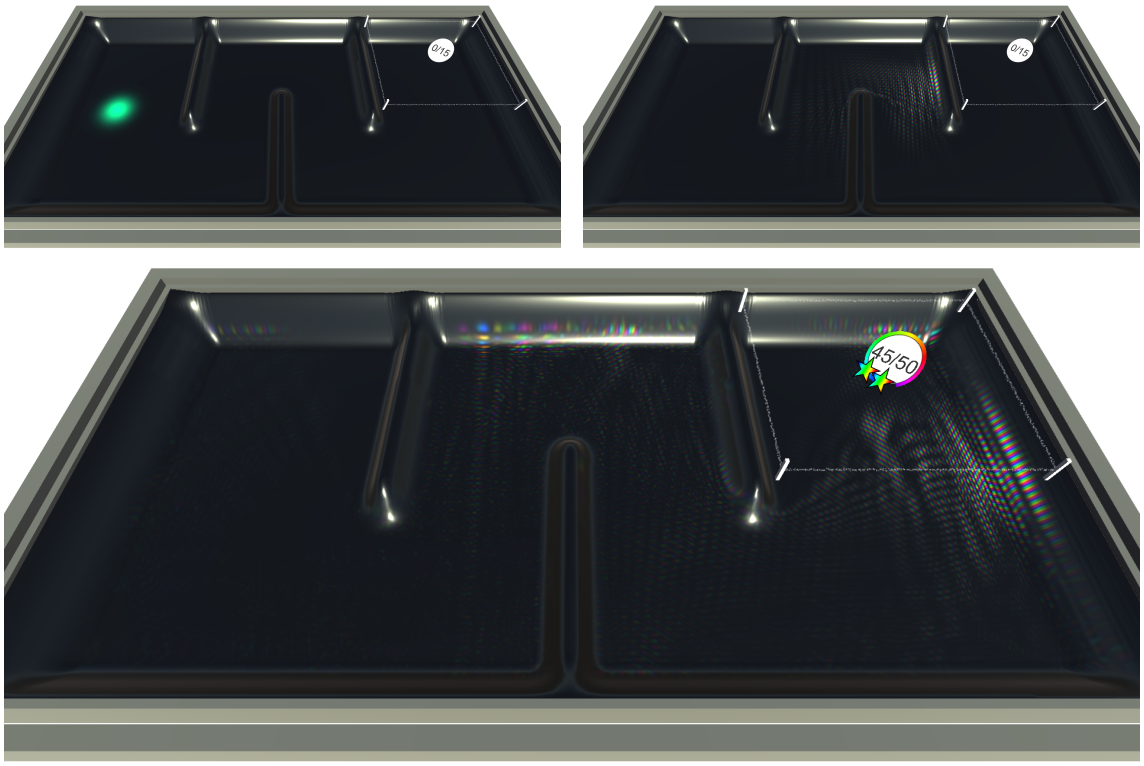


Figure 4.12: Level 2: A zigzag level

4.5.2 Level 3: Uncertainty in position

In Levels 3 and 4 the main feature is the split goal area. In order to beat these levels, the player must split the wave function into two parts.

This emphasizes that the uncertainty in position, intrinsic to quantum mechanics, can lead to situations where the particle seems to be in two places at once. As previously discussed in Chapter 2 this is not *actually* the case. Rather the particle has roughly equal probability to be found either in the upper half of the labyrinth or the lower half of the labyrinth.

Apart from introducing this concept the only challenging aspect of the level are the two asymmetric obstacles that need to be circumvented.

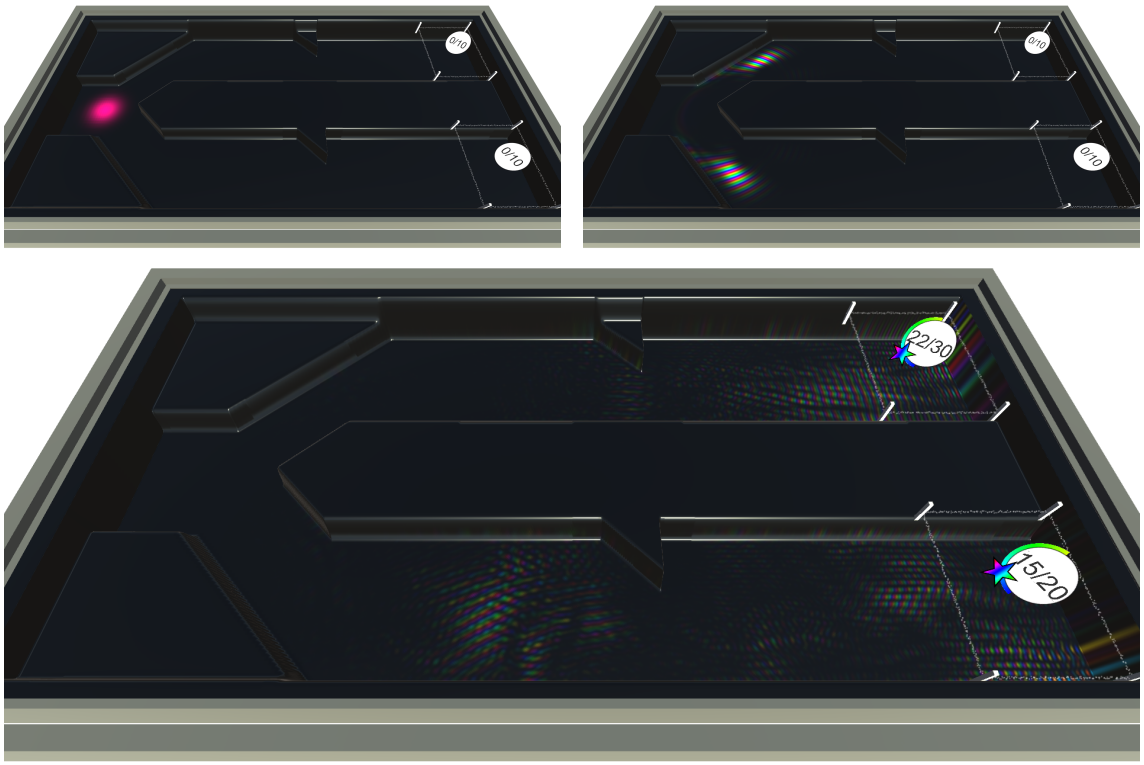


Figure 4.13: Level 3: Being in two places at once

4.5.3 Level 4: Measurement

Building upon the mechanics of Level 3, Level 4 introduces the concept of measurement and its impact on the wave function. Similar to the preceding level, the particle has to be *split* to fulfill two target areas. However, after reaching these areas the level doesn't end. Instead, a security system is triggered and the level continues.

The security system will do a measurement of the two halves of the labyrinth. Then the particle is found to either be in the upper half or lower half of the level. Afterward, the player has to adjust to the new situation and continue to steer the wave function to the goal area. Similar to the last level, a small ledge prevents the player from easily continuing on. Instead, depending on the outcome of the measurement, the player has to react and steer the wave packet around the obstacle.

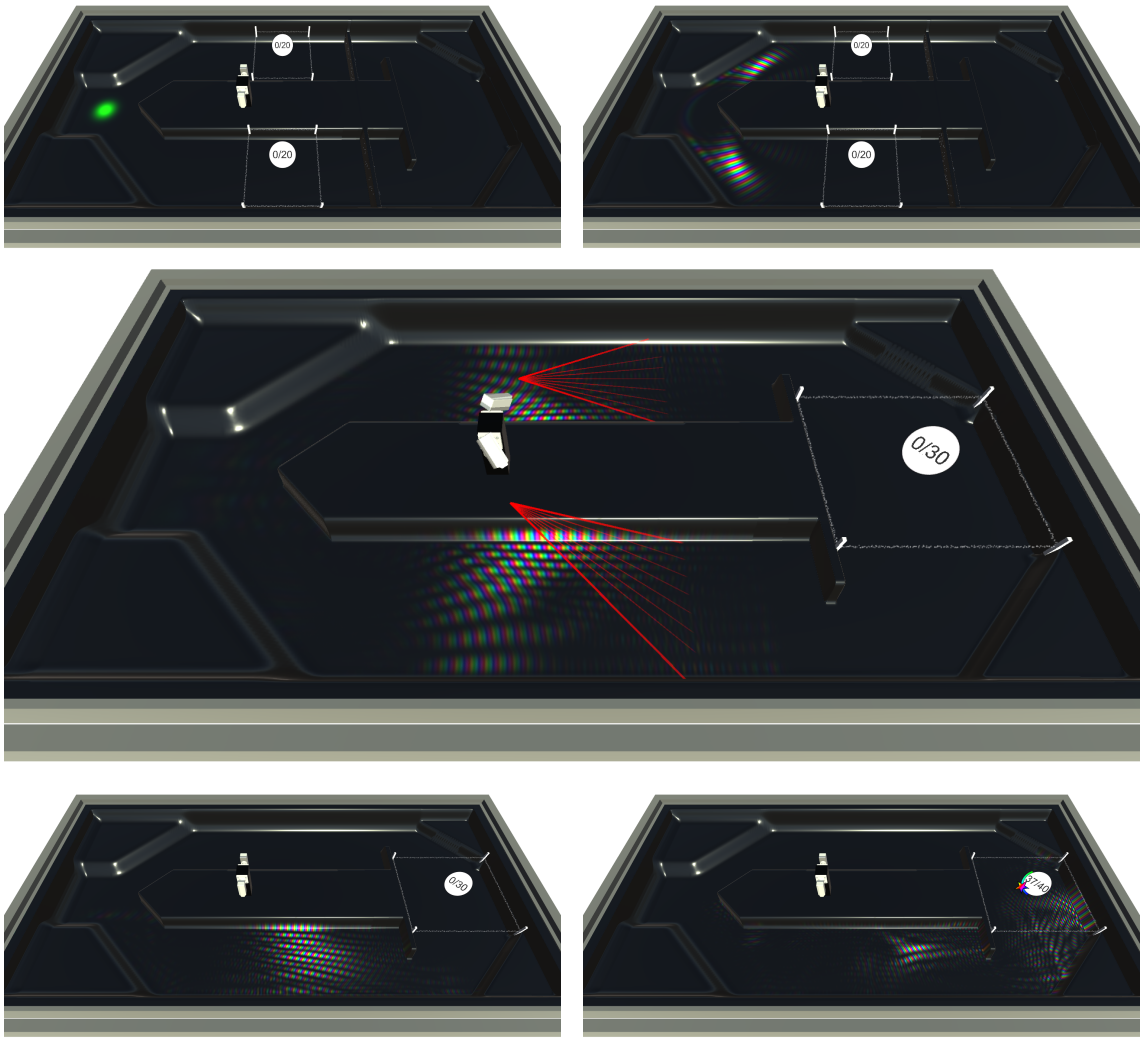


Figure 4.14: Level 4: Handling measurement

4.5.4 Level 5: Quantum tunneling

This level introduces the idea of tunneling to the player. There is a rather long path through the level, filled with many obstacles. While it might be possible to use that path, it is extremely difficult to reach the target with enough probability mass. Instead, an alternative solution is to build up speed and tunnel through a thin wall. It is thin enough to allow a decent amount of the wave function to continue past the wall, provided the wave packet has built up some energy. The most effective way to achieve this is to reflect off this wall, in order to go back and build up enough kinetic energy to tunnel through.

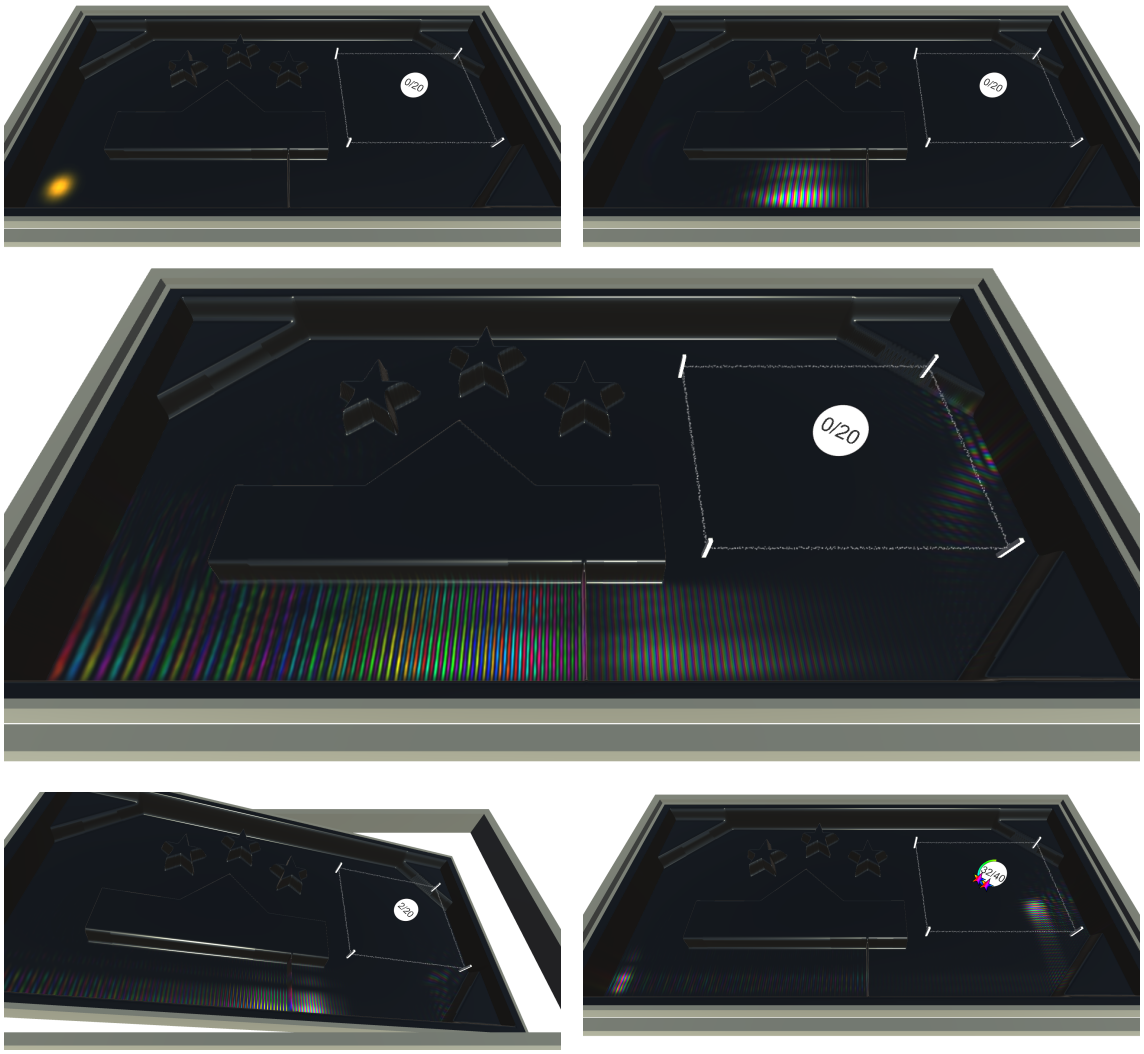


Figure 4.15: Level 5: Tunneling through a barrier

4.5.5 Level 6: Transmission and resonance of the finite well

In this level, the effects described in Section 2.2.3 have to be utilized in order to beat the level. This is also the only level in which the wave packet starts with some momentum.

In order to pass the first *ditch*, the wave should be slowed down to match the resonant frequency. It is also possible to speed up to pass this obstacle, however in order to pass the second obstacle the wave packet has to be slowed down. Otherwise, it will scatter while passing through the second ditch, making it almost impossible to gather enough probability density to beat the level.

If the wave packet gets slowed down enough it can reflect off the second triangle-like *ditch* and be easily steered to the target area. Since the wave function might be more spread out in this level than in the previous ones, it can be difficult to judge when to rotate the labyrinth.

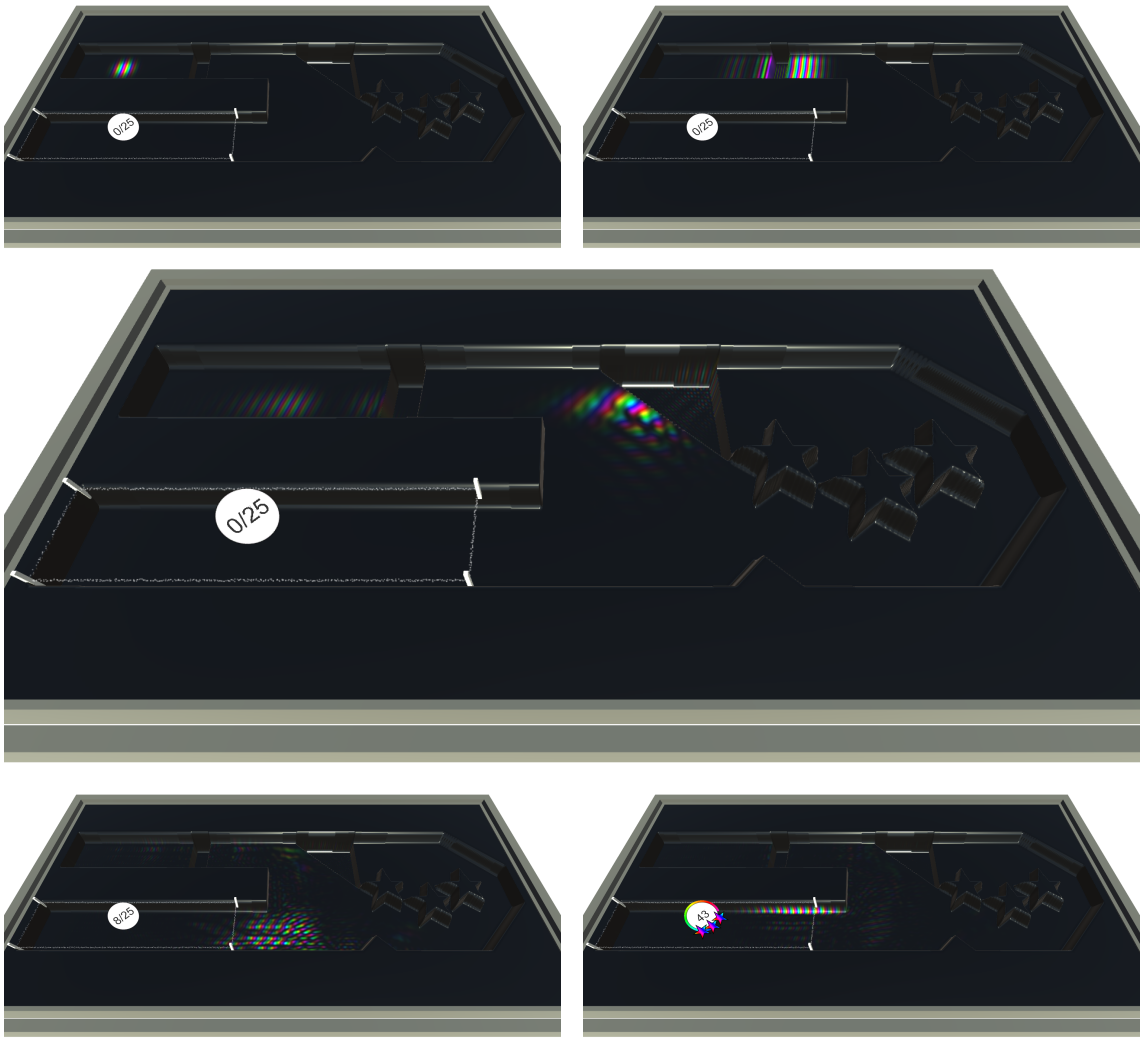


Figure 4.16: Level 6: Resonance and scattering in the finite well

4.5.6 Level 7: Interference patterns of the double slit experiment

Inspired by the very famous double-slit experiment, this level features two walls with small gaps. After passing through the slits of the first wall, the generated interference pattern has to be steered correctly to meet the gaps in the second wall. Otherwise, the parts that interfere constructively will reflect off the second wall and the level cannot be completed.

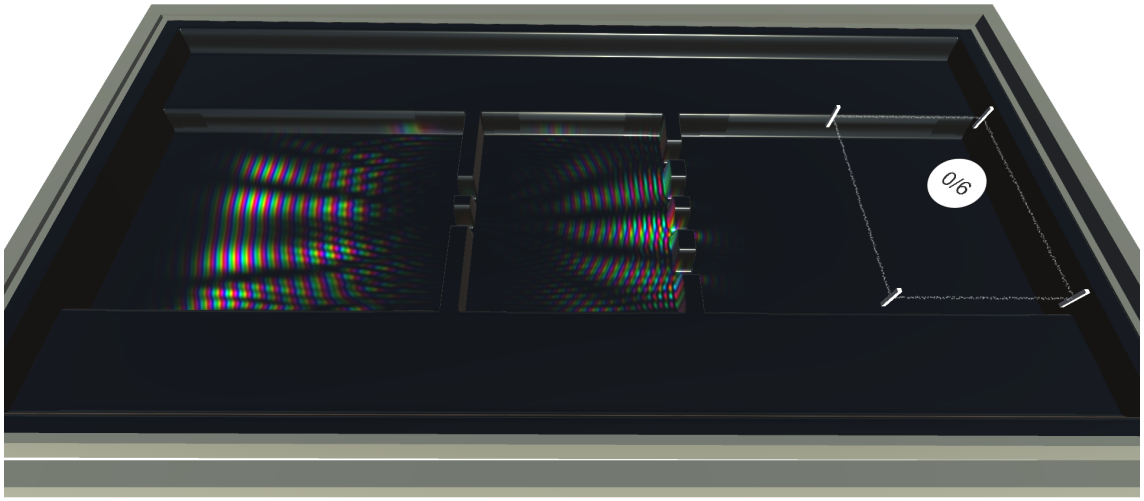


Figure 4.17: Level 7: Interference pattern after passing through the double-slit

As is the case with most other levels, reaching the minimum score is not designed to be terribly difficult. Reaching the full rating of 3 stars requires some more thought. For example, the player might choose to accelerate slower in the beginning to generate a simpler interference pattern, which can be steered more easily through the second wall.

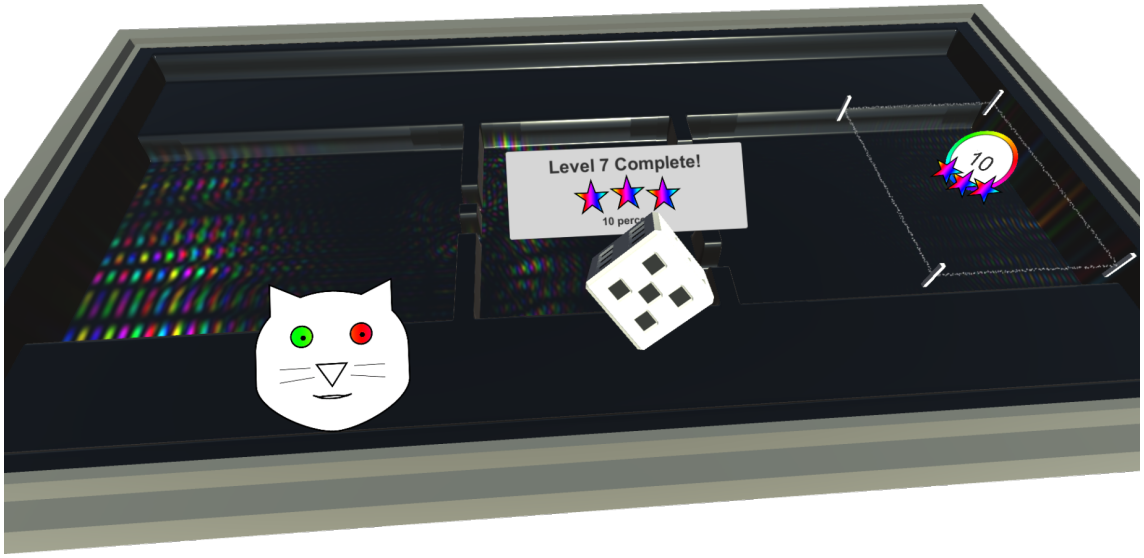


Figure 4.18: Level 7: Outro

Since this is the final level, there is a longer dialog after completion. The cat calculates the complete probability of being inside the target areas for all levels, once again reminding the player of the uncertainty which is inherent to quantum physics. Of course, the calculated chance will always be quite low, representing this reliance on chance, the camera slowly zooms out while a die rolls into frame, balancing on its edge.

4.6 Game engine

As previously mentioned, the game is realized using Unity[8]. The Unity game engine is very popular and there are many open-source projects and assets available. Furthermore, it supports the use of compute shaders which are used for the simulation.

While the engine theoretically supports many different platforms, including support for building HTML5 web applications, the reliance on compute shaders for the simulation limits the platforms that "Schrödingers Labyrinth" can be compiled to. While the game was developed and tested on *Windows*, Unity allows compilation on *Linux* or *MacOS* as well. In any case, the game requires a GPU that is capable of running the compute shaders. However, most GPUs are capable of running compute shaders[9].

The game itself is rather conservative regarding simulation size in order to allow as many systems as possible to have a smooth gameplay experience. As such the game can be played even on some integrated graphics chips. While the targeted minimum framerate is 30 frames per second, framerates can easily reach more than 100 FPS when playing the game on a computer using a dedicated GPU. This headroom in performance could be utilized to extend the game with more complex levels.

4.6.1 Potential function creation

The potential field $V(\vec{x})$ of the simulation is represented internally as a single channel texture. As mentioned in Section 4.4.2, it is used for the underlying simulation as well as to create the heightmap that gets used by the Unity terrain engine. Since the potential function is a simple texture, we can directly create a level's potential field using an ordinary image editor.

However, in order to simplify level creation, the potential field texture can be composited dynamically during runtime. This allows us to create commonly used level elements like walls and move them in the editor. For the more simple elements like walls or triangles, a shader was implemented that can generate these shapes in any size at runtime, vastly improving the level editing experience. Some other components, like the star shape in level 5, are simple textures that get layered on top of the potential.

Another advantage of this approach is, that parts of the level geometry can dynamically change during gameplay. This is used in level 4, where parts of the level are blocked with a wall that disappears once the security system is triggered. (See Section 4.5.3)

Since recomposing the potential field is a rather costly operation, the potential field should not be changed very often during gameplay to ensure good performance.

[8]Unity game engine: `https://unity.com/`
[9]Compute shader requirements: `https://docs.unity3d.com/Manual/class-ComputeShader.html`

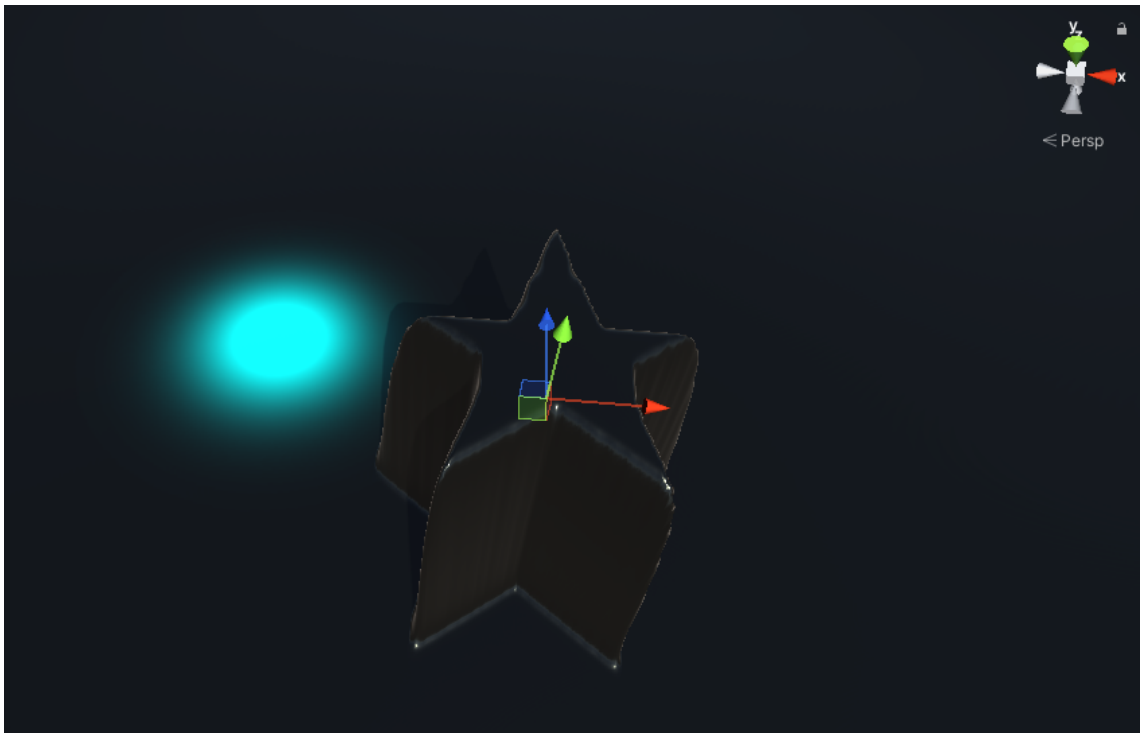


Figure 4.19: Star-shaped potential in editor

4.6.2 Tilting of the labyrinth

The user controls the game by tilting the labyrinth. This affects the simulation by actually tilting the potential field. This was implemented by applying a linear offset to the potential field, depending on the current rotation of the labyrinth. At their extreme points, the potential field might be adjusted by the player by up to $1eV$, each axis contributing up to $0.5eV$.

This means the when sampling the potential field, the value of the potential field at a given position (x, y) is calculated as follows:

$$V(x,y) = scale \cdot \left(P(x,y) + 0.25 \cdot r_x \cdot \frac{2x - X}{X} + 0.25 \cdot r_y \cdot \frac{2y - Y}{Y} \right) \tag{4.1}$$

With P being the single-channel potential texture, ranging from 0 to 1. The scale of the potential is $scale = 2eV$ and the rotation of the potential is described with r_x and r_y ranging from -1 to 1. The values X and Y describe the size of the simulation.

4.6.3 Implementation of the potential function visual representation

The potential is visualized using the unity terrain feature. The unity terrain creates a highly optimized mesh from a heightmap. In this case, the heightmap is a representation of the potential function. Using a heightmap as a source for the potential and its visual representation is very flexible and allows even complex potential functions to be implemented in the game.

Problems with the unity terrain implementation

There are some drawbacks and workarounds needed when dealing with the Unity terrain implementation.

The terrain's heightmap needs to adhere to some rules. First of all its resolution has to be $2^{n+1} \times 2^{n+1}$ which is not a big problem, since the Fourier transform also requires

its texture size to be a power of two. Also, the maximum value of the heightmap cannot exceed 0.5.

Both problems are dealt with by generating the heightmap from the potential texture rescaling and interpolating the last pixel as necessary. (Since the potential is of size 2^n and the heightmap 2^{n+1})

Another limitation of the unity terrain implementation is that it cannot be rotated, since it was designed to be used for a static world. But to tilt the labyrinth, rotation of the terrain is needed. In order to work around this limitation, the action of rotating the labyrinth is inverted. This means that the labyrinth itself is not actually rotated, instead the world, including the camera and skybox, is rotated around the labyrinth.

4.7 Objective scoring

The objective of the player is to reach the goal zone. Since the player does not control a singular classical character but rather a wave function, the objective works a bit differently. Scoring in each level is based upon the probability mass that the player managed to steer into the goal zone(s). Depending on the level the minimum scores are adjusted accordingly.

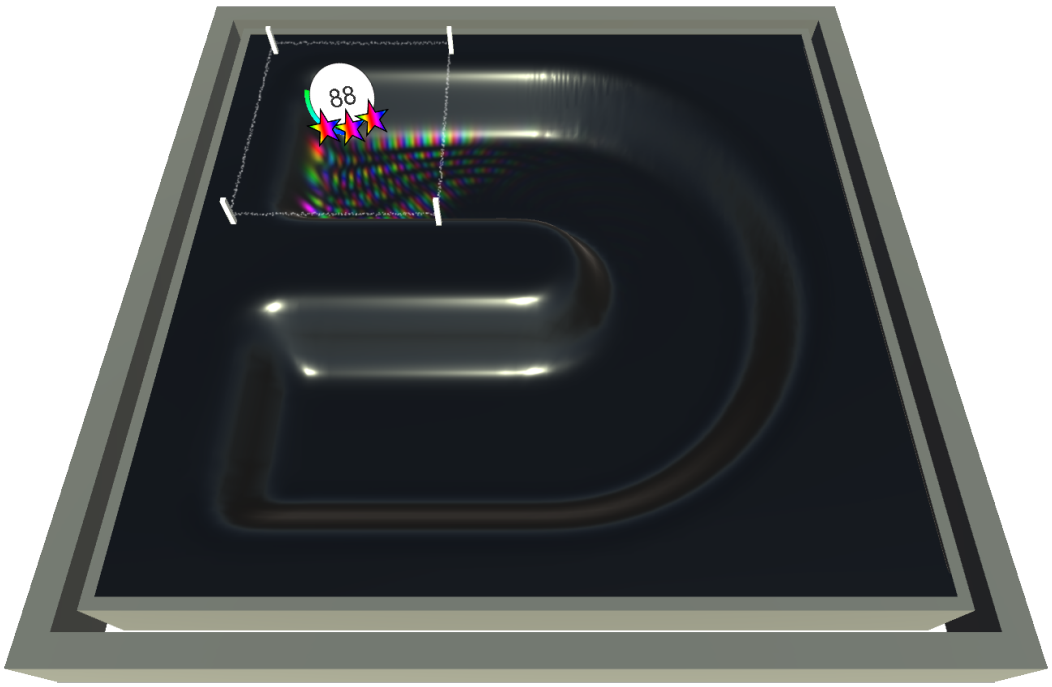


Figure 4.20: Scoring of the wave function

While the intro level allows the player to move almost the complete wave function inside the target area, later levels are more difficult. In most levels, it is almost inevitable to lose some of the probability mass to scattering in the environment.

$$\text{Score} = \int_{zone} |\Psi(x)|^2 dx$$

For example, level 5 requires the player to tunnel through a wall, which will lead to most of the wave function reflecting. As a consequence the target score is much lower, requiring only 20% of the wave function's probability mass to reach the target area.

After reaching the target area(s) the game does not end immediately. Instead, the game continues for 3 seconds, to give the player a chance to bring even more of the wave function

into the goal area. This grants a better score, with each level granting up to three stars at different minimum scores. The bonus score system provides players with an immediate idea of how good their score is. Additionally, the total score is kept to allow tracking high-scores even when reaching 3 stars in a level.

Since the score represents the probability mass that the player was able to move into the goal area, it also represents the probability of the cat reaching the exit. After finishing the last level this is picked up again by combining the scores of all levels to calculate an overall probability for the cat to escape. Of course, the combined probability will always be quite low since the probabilities get multiplied, but it serves as a good reminder that the position of a particle is not definite.

4.7.1 Implementation of the goal zones

The level's goal zones have to measure the probability mass inside them. For this purpose, a shader based on the already mentioned Prefix Sum shader[10] is used to do this efficiently on the GPU.

$$P_{zone} = \sum_{x=x_{min}}^{x_{max}} \sum_{y=y_{min}}^{y_{max}} |\Psi(x,y)|^2 \tag{4.2}$$

Since this calculation is a bit expensive, only one goal zone is allowed to update each frame, spreading the load over multiple frames when more than one goal zone is active at once. The calculation of goal zones is the most expensive operation apart from the actual simulation itself since the prefix sum algorithm requires multiple passes [LF80].

4.8 Sound design

The game's ambiance is supported by a calm background soundtrack. In an effort to give a better understanding of the current state of the wave function an ambient sound, whose pitch depends on the average energy of the wave function, provides auditory feedback of what is happening on the screen. The average energy of the wave function is calculated using a weighted sum of the amplitude of the momentum.

The level where this feature can be observed the easiest is level 6. The wave packet will increase in energy when entering the *ditch* that is featured at the beginning of this level. The increase of energy is directly reflected by increasing the pitch of the ambient sound.

The speaking cat's dialogs are spoken in by Daniel Tauritis, who kindly agreed to be part of the game. Having spoken dialogs, allows the given explanations to be much less dry and adds to the atmosphere of the game.

Most of the game does without any intrusive sound effects. The only exception being level 4, with its security system. During this level, some sound effects are used to better communicate that a measurement is happening.

[10] PrefixSum shader: `https://github.com/walbourn/directx-sdk-samples`

5 Experimental evaluation

In order to evaluate how the design goals of the game work in practice, a two-part questionnaire was used. Due to the Covid-19 pandemic, these play-tests needed to be carried out online only.

5.1 User survey design

The questionnaire is split into two parts. Before the game is played the participants are asked to estimate their knowledge about quantum mechanics and the Schrödinger equation. Since the game is targeted at people interested in quantum mechanics, we do not ask for specifics here, rather relying solely on the self estimate.

After playing, the participants are asked to evaluate the game regarding various aspects. In order to estimate how well the game teaches the various effects of quantum mechanics that are demonstrated in-game, a second set of questions presents various problems, asking the participants to predict the effects of quantum mechanics.

When asking the questions, we try to only require knowledge that was presented in the game. Since we do not introduce players to a lot of technical terminology, these questions have to be relatively simple.

5.2 Results

In the following, we present the results of the user survey. In total 18 participants played the game and answered the questionnaire. While this amount of participants does not allow us to do a very in-depth analysis of complex correlations between their answers, it can provide us with a good idea of how the game was received, and more importantly how it might be improved.

5.2.1 Before playing

At the beginning of the questionnaire, which takes place before the game is played, participants were asked about their background regarding quantum physics.

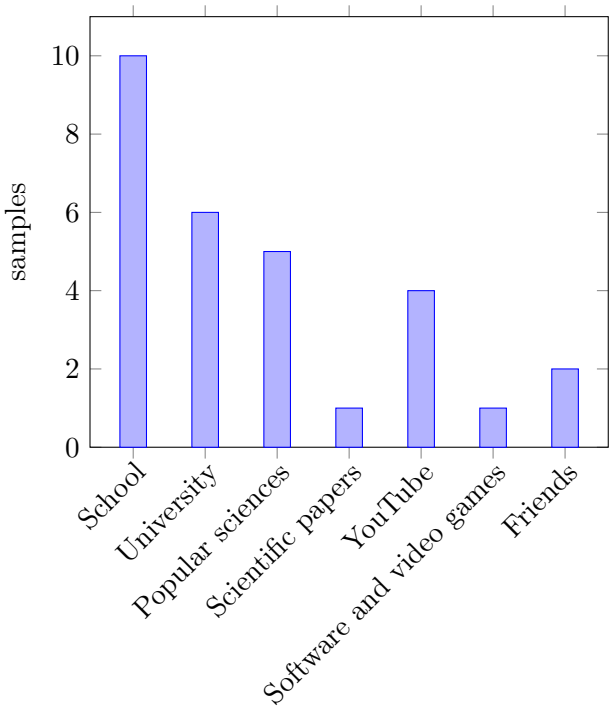


Figure 5.1: Source of knowledge

In general, as Figure 5.1 shows, a large portion of participants answered to know some amount of quantum mechanics from school or university, with *YouTube* and Popular sciences being very popular sources of knowledge too. Notably, only one participant claimed to have learned something about quantum mechanics from video games or other software.

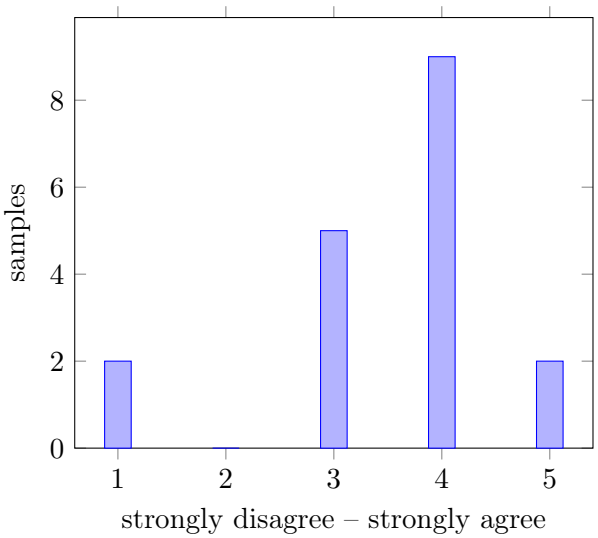
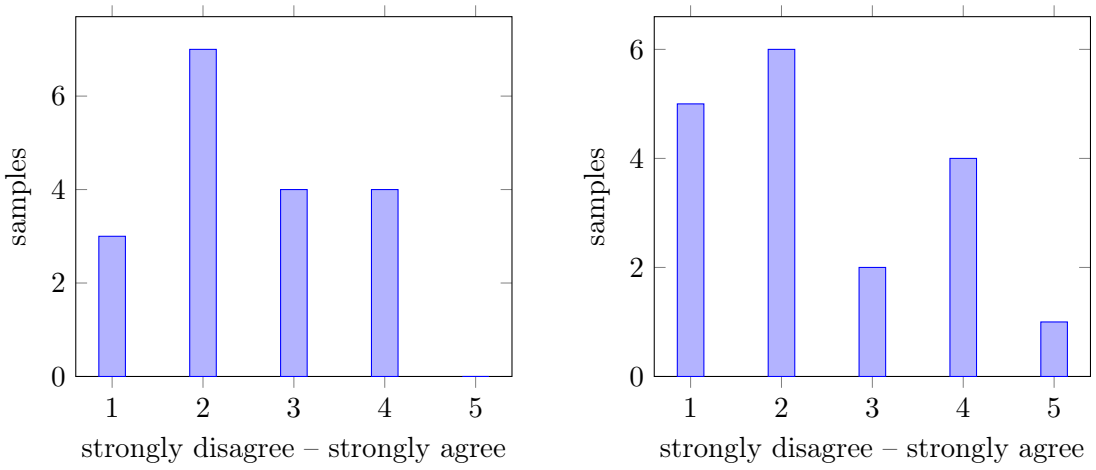


Figure 5.2: "I am interested in quantum mechanics"

As you can see in Figure 5.2, most participants of the survey were quite interested in quantum mechanics. Since quantum mechanics is a notoriously unintuitive area of physics, Figure 5.3 shows that most people estimate their knowledge in quantum mechanics to be rather low, even including some students that learned about quantum mechanics in university.



(a) "I know quantum mechanics" (b) "I know the Schrödinger equation"

Figure 5.3: Knowledge in quantum mechanics

Even fewer participants claimed to know the Schrödinger equation. Of course, that is to be expected since a lot of people source their knowledge of quantum physics from popular science and *YouTube* (see Figure 5.1), which usually put less focus on the Schrödinger equation.

5.2.2 After playing

After playing, participants were asked another two sets of questions. The first set of questions dealt with the overall impression of the game.

One thing that was very clear from the results after playing, is that the game is rather difficult, at times leading to frustration, which impacted how enjoyable the game is to play. This can also be observed clearly in how many participants were able to finish the game. (See Figure 5.4)

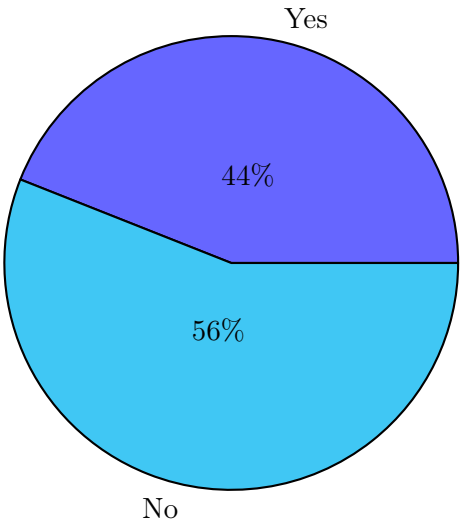


Figure 5.4: Amount of players completing every level

As you can see in Figure 5.5 the enjoyment participants got playing the game varies quite a bit. Going off the free-form feedback that players provided, the most frustrating parts of the game stem from the high difficulty paired with the slow speed, which makes replaying levels less interesting.

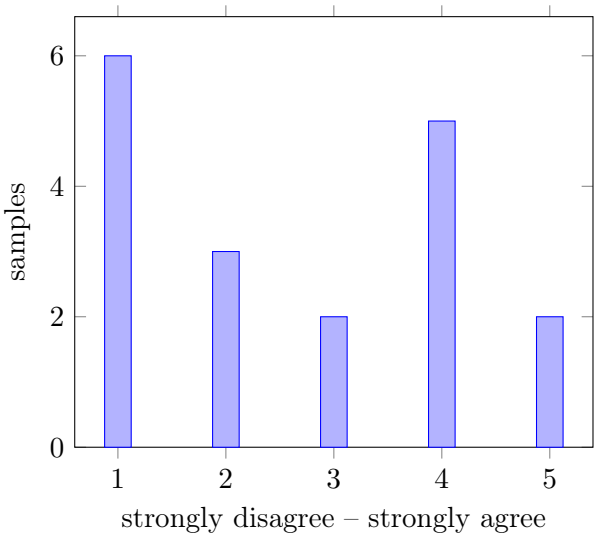


Figure 5.5: Fun playing the game

Some participants also found the available explanations given at the start of each level insufficient. Besides extending these explanations to be more in-depth, giving the player more hints during the game seems most promising. Note that for some levels hints are provided already, after restarting the level. However, considering the high difficulty of the game, it seems to be beneficial to implement such hints in every level and possibly extend them to show while playing a level too.

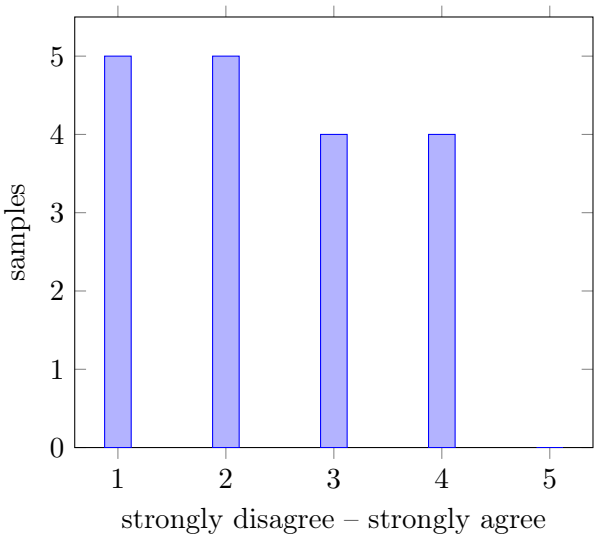


Figure 5.6: knowledge increase after playing

Another indication that the given explanations were sometimes lacking depth can be seen in Figure 5.6. Participants mostly stated that they felt rather little improvement in their understanding of quantum mechanics. To a certain degree, this might also be a consequence of the high difficulty, which meant that some levels were left unsolved.

In the next chapter, we will take a closer look at how well participants were able to predict certain phenomena of quantum physics after playing.

5.2.3 Intuition of quantum mechanics

In this chapter, we will take a look at how well participants were able to answer the second set of questions that were asked after playing. These questions were more concrete and asked about some of the quantum mechanical effects that could be observed in the game.

The questions covered the following topics:

- Uncertainty in position
- Uncertainty in speed
- Time evolution of a wave packet
- Correlation between the speed of a wave packet and its frequency
- Resonance in the finite well
- Interference patterns of the double-slit experiment

Uncertainty of position and speed

Figure 5.7 shows how many participants thought that particles in quantum mechanics generally have definite position or speed. As mentioned in Section 2.1.1 this is generally not the case, since the particle will usually be in a superposition. Furthermore, the eigenstates of position and momentum will never occur in reality, as they are not normalizable.

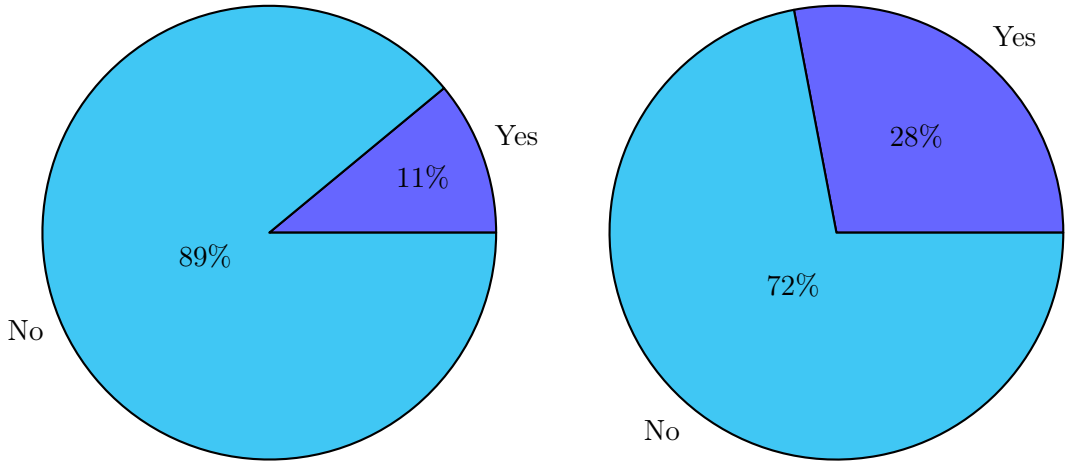


Figure 5.7: Does a particle have definite position (left) / speed (right)?

Most participants answered these questions correctly, however, there is a notable discrepancy between these two questions, with more participants answering that a particle has definite speed. This might be explained by the in-game representation of the wave function. While it is easy to see that the particle is *spread out* during the game, it is much harder to see that this is due to differences in speed. In particular, the average speed of a wave packet will be constant even though that does not mean the particle's speed has a definite value before measurement.

Time evolution of a wave packet

This theme carries over into some of the other questions too. When asked about how a wave packet will evolve in a constant potential (see Figure 5.8) most participants correctly observed that it will spread out over time. However, predictions involving the speed of the wave packet were a lot less accurate. The average velocity of a wave packet will not

change in a constant potential, nonetheless, a significant portion of participants answered that the wave packet will increase in speed.

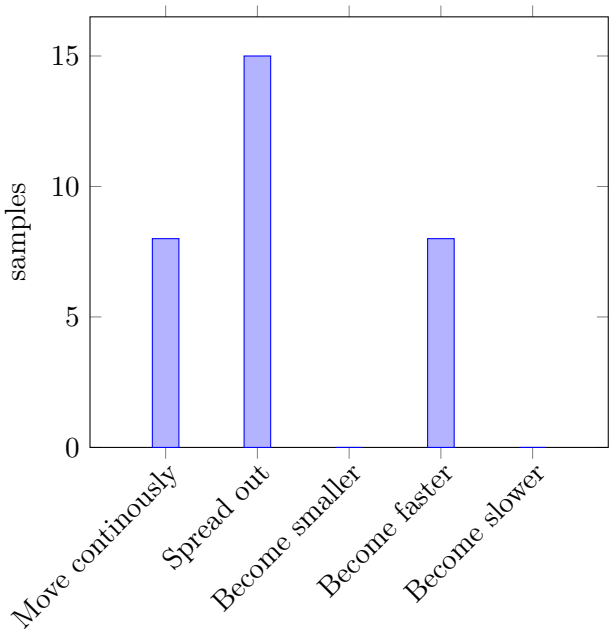


Figure 5.8: Evolution of a wave packet

When playing "Schrödingers Labyrinth", the wave packet will very often increase in speed, since the potential will usually not be constant, but instead be sloped by the player's input. These effects could potentially be mitigated by improving accompanying explanations, as well as including easier levels, where players can better get used to how the wave function evolves in simple environments.

Frequency of a wave packet

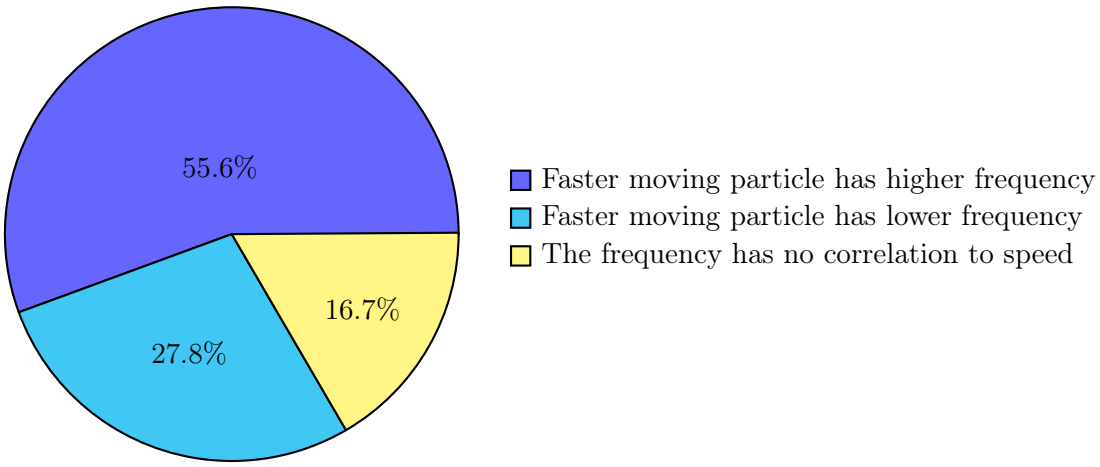


Figure 5.9: Correlation between frequency and speed of a wave packet

When asked about the frequency of the wave function (see Figure 5.9), slightly more than half of the participants correctly identified that a faster moving wave packets phase changes at a higher frequency. Since this correlation is not commented on in the game it is not surprising that a significant amount of players were not able to correctly answer this question. Similar to before it might be possible that including additional easier levels, can improve this result.

Resonsance of the finite potential well

There are two questions aimed at the effects of quantum mechanics specifically demonstrated in levels 6 and 7. In the first question, we asked participants to predict what can happen to a wave packet when moving towards a *potential ditch*. The expected effects were discussed in Section 2.2.3.

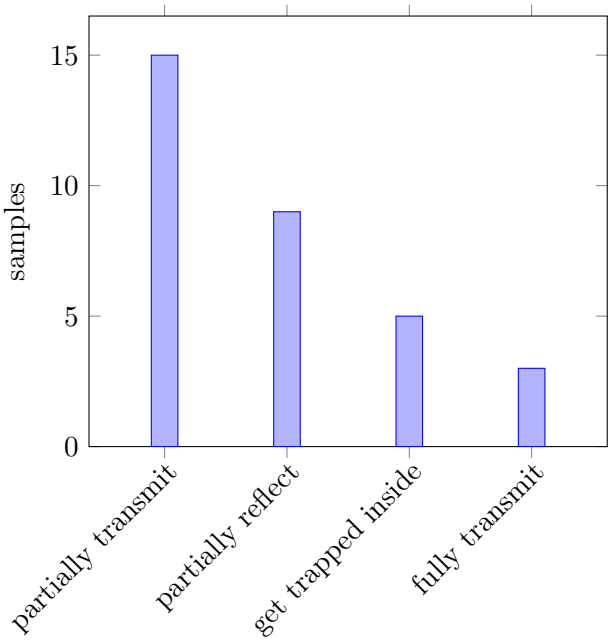


Figure 5.10: Transmission through a potential well

The results that are shown in Figure 5.10 clearly indicate that most participants were not able to identify the possibility of full transmission from playing the game. Since it is not required to fully transmit through the first ditch in order to beat level 6 and a wave packet strictly speaking will never fully transmit, because of uncertainty in momentum, this result is actually not completely surprising. Nonetheless, the game can provide a good demonstration of this phenomenon when paired with a deeper explanation of the underlying theory.

More interestingly only about half of the participants stated that the wave packet might partially reflect. When the player encounters the first *potential ditch* in level 6, it is possible to completely transmit by accelerating. However, as discussed in Section 4.5.5, the following level geometry is supposed to require a much slower wave packet. A slow wave packet will easily reflect off the first *potential ditch* when it is not the perfect speed to transmit due to resonance.

This might indicate a flaw in the level design of level 6, allowing players to beat it even when accelerating the wave packet. Considering that most participants were unable to beat every level, another possibility is that this level was skipped very often. Potentially improving accompanying dialog and hints of this level would yield better results.

It is also noteworthy that some participants answered that part of the wave packet might get trapped inside the *potential ditch*. While this is not possible when the potential field is constant, it might be possible in the game, as players tilt the labyrinth, which might allow them to capture some of the wave function inside the *potential ditch*.

Interference pattern of the double-slit experiment

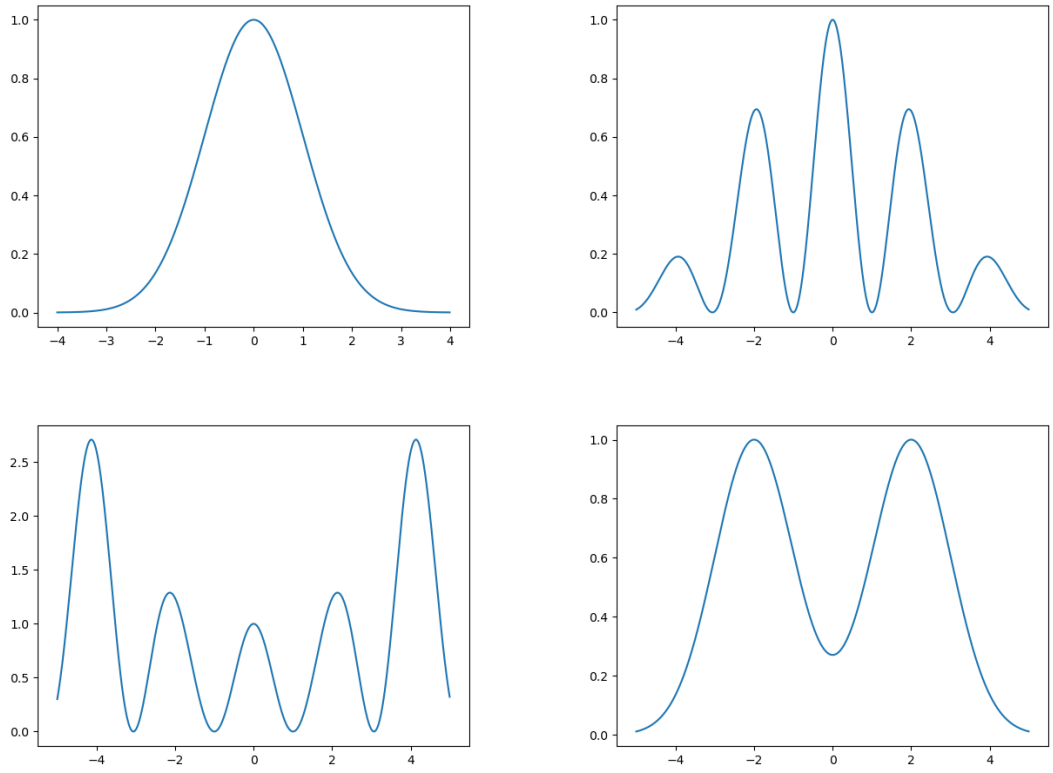


Figure 5.11: Probability density interference pattern of the double slit. Possible options going from top-left to bottom-right.

In the last question, participants were asked to predict the interference pattern of the double-slit experiment, similar to level 7 (see Section 4.5.6). Most player correctly identified the correct interference pattern (top-right in Figure 5.11), with some picking the bottom-right option instead.

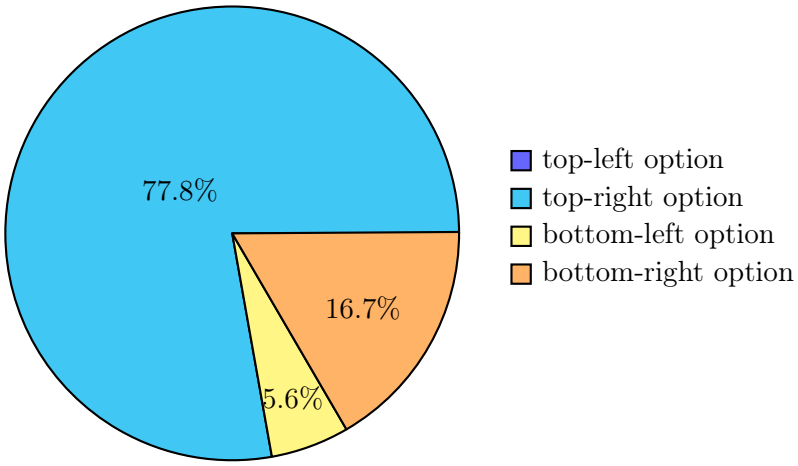


Figure 5.12: Interference pattern of the double slit experiment

Level 7 seems to be quite successful in demonstrating the expected interference pattern. In this case, we need to keep in mind that the double-slit is a very popular experiment, so

in contrast to the effects presented in level 6, a lot more people might already know this interference pattern.

5.2.4 Estimating prior knowledge

Since the participants have very different prior knowledge about quantum mechanics (as discussed in Section 5.2.1), it can be a bit difficult to correctly interpret the results of Section 5.2.3. We purposely opted not to ask these knowledge questions before playing, to not overwhelm participants who are not particularly familiar with quantum physics.

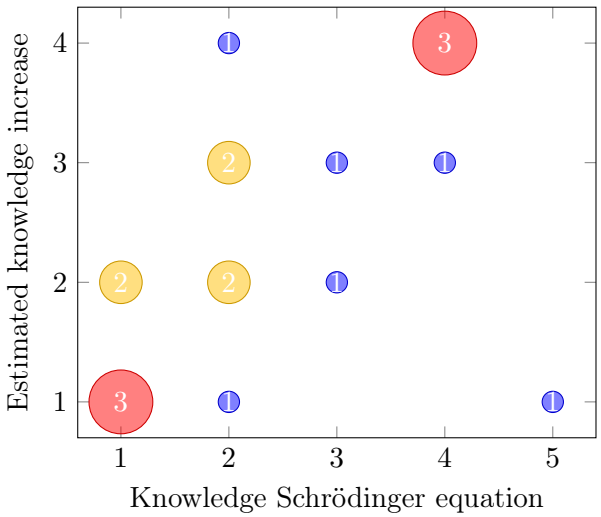


Figure 5.13: Correlation between prior knowledge of the Schrödinger equation and increased knowledge after playing

Instead, we rely solely on participant's estimates of their knowledge regarding quantum mechanics in general and the Schrödinger equation specifically. When analyzing these estimates there is no clear correlation between estimated prior knowledge of quantum mechanics in general and the estimated increase in knowledge after playing. However, plotting knowledge about the Schrödinger equation against the estimated increase in knowledge after playing, people who knew the Schrödinger equation beforehand seem to generally think, that their understanding of quantum mechanics increased after playing.

This might indicate a rather high barrier of entry, making the game more suitable for players with a good theoretical understanding of quantum mechanics.

5.3 Possible improvements

Considering the results of the survey, and especially the free-form feedback, we will discuss some possible improvements.

The most apparent problem is the high difficulty of the game with only 44% of participants completing every level. In order to combat this issue and to make it easier to understand what is going on without having good background knowledge on quantum mechanics, accompanying dialogs need to be extended. Also providing more hints during gameplay would be beneficial.

Possibly most important, the player needs more time to get used to controlling the wave function. This means additional (simple) levels to introduce the player to the game might allow more players to finish the game, and thus learn more about the actual background.

In order to extend the game, the most important improvement is an increase in the level's length and complexity. Since the performance and accuracy of the simulation are

quite good, there is still some headroom to build more complex levels, which incorporate more elements of the existing levels in new combinations.

A point that was mentioned multiple times, is the slow speed of the simulation. There are some problems with increasing simulation speed in general. Since the wave function has a phase, increasing simulation speed will also increase the frequency with which the phase of the wave function changes.

However, implementing a fast-forward function, that switches the in-game view to a representation without phase, is a suitable improvement. This would allow the player to fast-forward parts of the level, which don't require complicated user input.

This function and some other minor improvements that were requested by participants were implemented and can be found in an updated build of the game.

5.3.1 Outlook

Building upon the foundation of the game, a very interesting extension would be the inclusion of a level editor. This could be used to recreate experiments and would give players the freedom to experiment with completely different level architectures. Since the game architecture already supports compositing the potential field from multiple sources (see Section 4.6.1), this extension could be easily integrated into the game as is.

In this context, a more simulation-like approach, which provides players with more actual numbers of the simulation (such as mass, or size), might be interesting. This could be particularly interesting for players that are already knowledgeable in quantum mechanics, such as students.

6 Conclusion

Quantum Mechanics is becoming increasingly more important, especially considering the advent of quantum computing. Still, it remains one of the most unintuitive areas of physics since things behave very differently from what we are used to.

There are lots of approaches to learning about quantum mechanics. While most people learn about quantum mechanics in school or university, sources like online videos and popular sciences are also quite popular. In contrast, learning with software and games only happens rarely. (See Section 5.2.1)

In this thesis, I presented "Schrödingers Labyrinth", a game about the Schrödinger equation, which tightly integrates time evolution of quantum mechanical systems as its core gameplay principle. Since the Schrödinger equation itself is a very theoretical part of quantum physics, even people who are interested in quantum mechanics might not have heard about it. Most games that teach quantum mechanics opt for a more indirect approach of showing quantum mechanical effects. "Schrödingers Labyrinth" instead puts the player directly in control over a wave function of a single particle.

The underlying simulation is implemented on the GPU using the split-step Fourier method, which allows complex potential fields to be simulated efficiently in order to demonstrate various effects of quantum mechanics. (See Section 3.4.2)

When playing the game, players are guided through several levels, which get increasingly more difficult. While progressing through the game, players learn visually how the wave function evolves over time and how it reacts to various obstacles. Over the course of the game, fundamental parts of quantum mechanics such as quantum tunneling, uncertainty, interference, and measurement are demonstrated. (See Chapter 4)

After playing, most players were able to successfully predict the more obvious effects demonstrated. However, some questions of the questionnaire that was completed after playing were significantly more difficult to answer. This is true especially for questions that were related to the speed or phase of the wave packet. In order to rectify this, more levels with more in-depth explanations and hints might provide a better learning experience.

When evaluating the game a clear problem seems to be the high difficulty, which could lead to frustration. In order to combat this, more hints and additional introduction levels might be necessary. Nonetheless, the "Schrödingers Labyrinth" provides a fluent real-time simulation of how a single particle in an arbitrary potential field behaves. This was especially well received by players already familiar with the Schrödinger equation, who appreciated the approachable visualization of time evolution in quantum mechanics. By extending the game with more levels and better explanations to make it more accessible to players with less background knowledge, "Schrödingers Labyrinth" could enhance the experience when learning about quantum physics, especially when paired with other media. A worthwhile extension of the game might also be the inclusion of a level editor to allow players to create their own levels in which they can observe how the wave function reacts.

Bibliography

[AES+03] Arnold, Anton ; Ehrhardt, Matthias ; Sofronov, Ivan u. a.: Discrete transparent boundary conditions for the Schrödinger equation: Fast calculation, approximation, and stability. In: *Communications in Mathematical Sciences* 1 (2003), Nr. 3, S. 501–556

[AEZ13] Adams, Allan ; Evans, Matthew ; Zwiebach, Barton: *8.04 Quantum Physics I. Spring 2013*. Massachusetts Institute of Technology: MIT OpenCourseWare https://ocw.mit.edu, 2013

[Bae00] Baer, Roi: Accurate and efficient evolution of nonlinear Schrödinger equations. In: *Physical Review A* 62 (2000), Nr. 6, S. 063810

[CN47] Crank, J. ; Nicolson, P.: A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 43 (1947), Nr. 1, S. 50–67. `http://dx.doi.org/10.1017/S0305004100023197`. – DOI 10.1017/S0305004100023197

[Dat10] Datta, Biswa N.: *Numerical linear algebra and applications*. Bd. 116. Siam, 2010

[DOST96] Dattoli, G ; Ottaviani, PL ; Segreto, A ; Torre, A: Symmetric-split-operator techniques and finite-difference methods for the solution of classical and quantum evolution problems. In: *Il Nuovo Cimento B (1971-1996)* 111 (1996), Nr. 7, S. 825–839

[FH65] Feynman, Richard P. ; Hibbs, Albert R.: *Quantum mechanics and path integrals*. New York, NY : McGraw-Hill, 1965 (International series in pure and applied physics). `https://cds.cern.ch/record/100771`

[FLS11] Feynman, R.P. ; Leighton, R.B. ; Sands, M.: *The Feynman Lectures on Physics, Vol. III: The New Millennium Edition: Quantum Mechanics*. Basic Books, 2011 (The Feynman Lectures on Physics). `https://www.feynmanlectures.caltech.edu/III_toc.html`. – ISBN 9780465025015

[FSS+09] Fehske, Holger ; Schleede, Jens ; Schubert, Gerald ; Wellein, Gerhard ; Filinov, Vladimir S. ; Bishop, Alan R.: Numerical approaches to time evolution of complex quantum systems. In: *Physics Letters A* 373 (2009), Nr. 25, S. 2182–2188

[Ful19] Fullerton, Tracy: *Game design workshop: a playcentric approach to creating innovative games*. AK Peters/CRC Press, 2019

[LBG08] Lloyd, D B. ; Boyd, Chas ; Govindaraju, Naga: Fast computation of general Fourier transforms on GPUs. In: *2008 IEEE international conference on multimedia and expo* IEEE, 2008, S. 5–8

[LF80] Ladner, Richard E. ; Fischer, Michael J.: Parallel prefix computation. In: *Journal of the ACM (JACM)* 27 (1980), Nr. 4, S. 831–838

[MPS10] Maheswari, A U. ; Prema, P ; Shastry, CS: Resonant states and transmission coefficient oscillations for potential wells and barriers. In: *American Journal of Physics* 78 (2010), Nr. 4, S. 412–417

[Neu27] Neumann, J. v.: Wahrscheinlichkeitstheoretischer Aufbau der Quantenmechanik. In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* 1927 (1927), 245-272. `http://eudml.org/doc/59230`

[NS99] Nettesheim, Peter ; Schütte, Christof: Numerical integrators for quantum-classical molecular dynamics. In: *Computational Molecular Dynamics: Challenges, Methods, Ideas*. Springer, 1999, S. 396–411

[Sch26] Schrödinger, E.: An Undulatory Theory of the Mechanics of Atoms and Molecules. In: *Physical Review* 28 (1926), Dezember, Nr. 6, S. 1049–1070. `http://dx.doi.org/10.1103/PhysRev.28.1049`. – DOI 10.1103/PhysRev.28.1049

[Sch05] Schlosshauer, Maximilian: Decoherence, the measurement problem, and interpretations of quantum mechanics. In: *Reviews of Modern physics* 76 (2005), Nr. 4, S. 1267

[Str68] Strang, Gilbert: On the construction and comparison of difference schemes. In: *SIAM journal on numerical analysis* 5 (1968), Nr. 3, S. 506–517

[Vis91] Visscher, PB: A fast explicit algorithm for the time-dependent Schrödinger equation. In: *Computers in Physics* 5 (1991), Nr. 6, S. 596–598

[WZS+14] Wang, Endong ; Zhang, Qing ; Shen, Bo ; Zhang, Guangyong ; Lu, Xiaowei ; Wu, Qing ; Wang, Yajuan: Intel math kernel library. In: *High-Performance Computing on the Intel® Xeon Phi™*. Springer, 2014, S. 167–188

[Zeh70] Zeh, H D.: On the interpretation of measurement in quantum theory. In: *Foundations of Physics* 1 (1970), Nr. 1, S. 69–76