Guitar Virtual Instrument using Physical Modelling with Collision Simulation

Conference Paper · July 2021

CITATIONS
0

READS
1,803

3 authors, including:

Chuck-jee Chau
The Chinese University of Hong Kong
20 PUBLICATIONS **204** CITATIONS

SEE PROFILE

Guitar Virtual Instrument using Physical Modelling with Collision Simulation

Ka-wing Ho* **Yiu Ling*** **Chuck-jee Chau**

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong

kwho@link.cuhk.edu.hk yling@link.cuhk.edu.hk chuckjee@cse.cuhk.edu.hk

* These authors contributed equally to this work.

ABSTRACT

We have created a guitar virtual instrument by simulating string vibration using a finite difference method to solve a modified one-dimensional wave equation with damping and stiffness, along with a collision system that allows the simulated guitar to perform a variety of articulations. Convolution with impulse response is also used to enhance the realism of the sound. The core model design, implementation approach and the optimization techniques are presented in this paper.

1. INTRODUCTION

Guitar is a very popular musical instrument that can produce tones with rich dynamics because of the variety of ways to interact with the strings. A small difference in finger motion can make a big difference in tones.

Sample-based synthesis is commonly used for existing commercial guitar virtual instruments. However, there may be difficulties in smoothly transitioning between articulations and providing fine-grained controls due to the number of samples required. Although multiple physical modelling schemes had been proposed [1], they cannot fully simulate the large range of articulations that can be performed on guitars. Therefore, we would like to develop a physical modelling scheme with a newly added collision system to emulate the way guitars are played so that better flexibility can be achieved for music production.

We have prepared a website[1] with demo videos and articulation animations to demonstrate our prototype program for practical uses.

2. STRING VIBRATION MODEL

2.1 Wave Equation

To model the vibration of a guitar string, we use a partial differential equation (PDE). The equation is modified from the 1D wave equation [2]:

$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial^2 y}{\partial x^2} \quad (1)$$

The 1D wave equation allows us to simulate the vibration of an ideal string where the motion is strictly confined to a 2D plane. This has the advantages of being less complex and more lightweight than simulating strings in 3D space.

Based on the presentation by Shuppius [3], we get equation (2) which accounts for stiffness as well as frequency dependent and independent damping factors:

$$\frac{\partial^2 y}{\partial t^2} = \frac{T}{\mu}\frac{\partial^2 y}{\partial x^2} - 2\sigma_0\frac{\partial y}{\partial t} + \sigma_1\frac{\partial}{\partial t}\frac{\partial^2 y}{\partial x^2} + EI\frac{\partial^4 y}{\partial^4 x} \quad (2)$$

To allow the tension of the string to be changed over time, we introduce a new variable tension factor *T(t)*:

$$\frac{\partial^2 y}{\partial t^2} = \frac{T(t)}{\mu}\frac{\partial^2 y}{\partial x^2} - 2\sigma_0\frac{\partial y}{\partial t} + \sigma_1\frac{\partial}{\partial t}\frac{\partial^2 y}{\partial x^2} + EI\frac{\partial^4 y}{\partial^4 x} \quad (3)$$

$T(t)$: tension of the string at time t

μ: linear density of the string

σ_0: factor for frequency-independent damping

σ_1: factor for frequency-dependent damping

E: Young's modulus of the string material

I: second moment of area $= \pi r^4/4$ for a cylinder

In equation (3), *y* is a function of *x* and *t*, representing the displacement of the string segments along the y-direction at position *x* at time *t*, see Fig. 1. By changing *T* and μ, we can set the pitch of the string, and by changing the damping and stiffness factors, we can modify the timbre.

2.2 Boundary and Initial Conditions

To allow our model to simulate strings fixed by the bridge and nut of the guitar, we define the boundary condition as:

$$y(l,t) = 0, where\ l \le 0\ or\ l \ge L \quad (4)$$

In our model, $x = 0$ and $x = L$ are assumed to be the position of the nut and bridge of the guitar respectively, where *L* is the length of the string.

As we will discuss in Section 3, collision is used for applying excitation to the string during the simulation, thus we can set the initial condition to be at the resting position:

$$y(x,0) = 0 \quad (5)$$

2.3 Finite Difference Method

It is difficult to analytically solve a complicated PDE such as the one we use. Collision and external force simulation also add additional complexity. Therefore, we choose to use a numerical approximation with a finite difference scheme based on that presented by Shuppius [3] instead.

[1] https://weonix.github.io/Physical-Guitar-ICMC-Demo/

We discretize the model by dividing the space domain along the string into nodes with distance Δx between adjacent ones. The time domain is also divided into discrete steps with interval Δt in between. Then we can define the temporal and spatial resolution as such:

$$spatial\ resolution = \Delta x^{-1} \quad (6)$$

$$temporal\ resolution = \Delta t^{-1} \quad (7)$$

We can then obtain our finite difference scheme by substituting each partial differential with a finite difference approximation. The formula we obtain consists of terms of $y(x \pm n\Delta x,\ t \pm m\Delta x)$, where m and n are non-negative integers. After moving the term $y(x,\ t + \Delta t)$ to one side, we get a formula that computes $y(x,\ t + \Delta t)$ using the $y(x \pm n\Delta x,\ t - m\Delta x)$ terms where $m \leq 0$. Hence, we get a stepwise algorithm that computes the new state of the string at each time step using the information of previous steps.

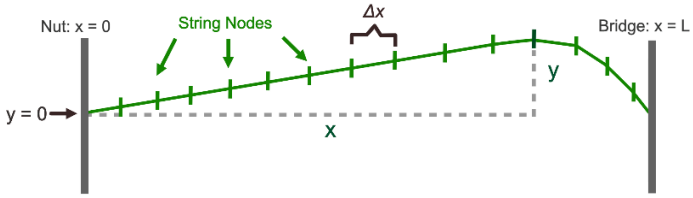

Figure 1. Coordinate system and node setup of the string model. (Not to scale)

2.4 Stability Condition

The stability of a finite difference scheme is determined by the Courant–Friedrichs–Lewy condition [2]:

$$C = \frac{u\Delta t}{\Delta x} \leq C_{max} \quad (8)$$

In our model, u is the string wave speed, defined by:

$$u = \sqrt{\frac{T}{\mu}} \quad (9)$$

A finite difference model can remain stable only if the Courant number C is less than or equal to a fixed value C_{max} which depends on the nature of the model. Through our experimentation, we have found that the C_{max} of our model is around 1. To ensure the model stability regardless of varying tension, we must also set an upper limit to the tension T as T_{max}. Thus, for a given setting, the maximum spatial resolution can be determined as such:

$$spatial\ resolution \leq temporal\ resoltuion \sqrt{\frac{\mu}{T_{max}}} \quad (10)$$

3. COLLISION

We also add collision simulation into our stepwise simulation algorithm. To simulate collision, we define a number of colliders in the scene. A collider can represent objects such as a guitarist's finger that may interact with the string, or a fret wire on the fretboard.

We perform collision by computing a vertical collision boundary based on the collider placements at each time step. Then if the y-axis displacement of a node exceeds its collision boundary, it will be forcefully set to the boundary, so that the string's motion will be obstructed by the colliders appropriately.

To improve the performance, we can also label the colliders as "static" or "dynamic" and do not recompute the collision boundary of static colliders at each simulation cycle unless they are moved explicitly.

3.1 Soft Collision

To better model colliders that are not completely rigid (e.g. a guitarist's finger and palm), two parameters: softness and elasticity are used to approximate this property.

To simulate softness, when a string node exceeds its collision boundaries, instead of setting its displacement value to be exactly at the collision boundary, it is set to be a ratio (i.e. the softness value) between the displacement and the boundary, so that the string can partially overlap with the collider before being pushed out over time.

Elasticity determines the energy loss that the string experiences when it overlaps with colliders. At each simulation step, if a string node is overlapping with a collider, the new displacement of the node will be moved to a position closer to its original displacement, essentially slowing down the node movement.

4. ARTICULATION

Using our collision model, we have a lot more flexibility in manipulating the string over the traditional method of simply setting initial conditions (e.g. a triangular shape). We make use of fixed colliders placed along the string to represent the fretboard. We can then use other colliders to press onto the string against the frets and to pull on the string before releasing, to simulate string fretting and plucking motions as in Fig. 2 and 3.

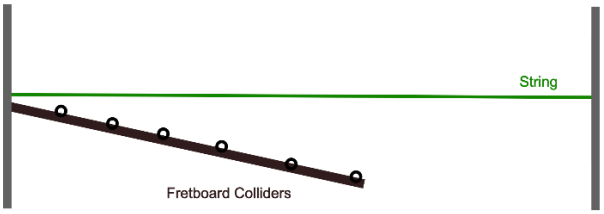

Figure 2. The fretboard arrangement of the string model at the resting position. (Not to scale)

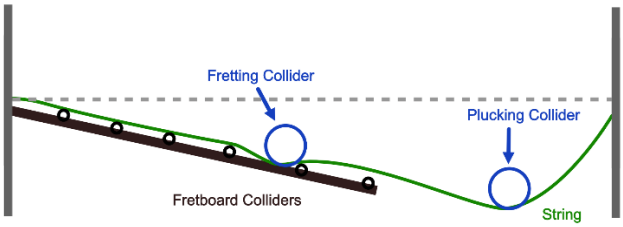

Figure 3. Fretting and plucking colliders acting on a string to perform a plucking mo. (Not to scale)

Because of the limitation of simulating string vibration on a 2D plane, when we place the fret colliders along the string, the string may hit the fretboard and produce

undesirable sounds. To alleviate this, we placed the colliders at a relatively steep slope, so that the unused frets are less likely to collide with the vibrating string.

Using our method, incidental noise caused by the interactions between the string and colliders, like the noise of a fretting finger being released, will be generated as a natural result, which enhances the realism of our model.

By using different combinations of collider motions and parameter changes, we can implement a variety of articulations used by real guitar performance, such as palm mute, tapping, slapping, harmonics, sliding, hammer-on and pull-off. Effects like fretted and fretless guitars or using fingers and plectrums can also be simulated. Some techniques used to reproduce these effects are listed below:

- **Bending and Vibrato:** The pitch of the note that is being played can be controlled by modulating the tension of the vibrating string, resulting in a pitch-changing effect.
- **Palm Mute:** We place a soft collider representing the palm pressed against the string near the bridge to absorb some of the string vibration and produce a muted sound.
- **Dead Note:** The fretting collider only presses softly against the frets while the frequency-independent damping factor of the string is increased so that the notes played become short and muted.
- **Slapping and Popping:** We reduce the slope of the fret colliders so that the string would be more likely to collide with the fret wires and fingerboard to produce a collision noise typical to the slapping and popping articulations.
- **Natural and Artificial Harmonics:** Before we play a note, we place an additional soft and non-elastic collider that overlaps the string at a specific harmonic ratio of the string length, which results in a unique harmonic tone.
- **Sliding:** We can simply slide the fretting collider along the string as a note is being played, so the length of the vibrating string and the pitch of the note will be changed.
- **Finger and Plectrum:** To simulate the difference between plucking with a finger and plectrum, we use a relatively soft collider to represent the finger, and increase the softness of the collider as we release the finger to emulate the string sliding off the soft flesh, which produces a softer and duller tone than the one with a hard plectrum.

5. AUDIO GENERATION

In order to generate audio from string vibration, we first take a sample of the displacement of the string at a specific position(s) after every simulation step (Δt). Then we can make use of the current and previous displacements to compute the velocity of the string at the sampling point(s), which will be used as the amplitude for audio output.

To simulate other elements that affect the guitar tone, such as acoustic guitar bodies, or properties of pickups and microphones, we take inspiration from a project by Harriman [4] and choose to use convolution with impulse response samples. This allows us to emulate a large range of guitar models and environments without much increase in the model complexity.

5.1 Magnetic Pickup

To simulate a magnetic pickup used for electric guitars, we can set the sampling position as the pickup position, which can be adjusted to produce different tones like in real electric guitars, and then we can apply convolution with an impulse response sample obtained from the real pickup to simulate the tone of this pickup. Multiples pickups can also be used at once by adding up the audio signals obtained from multiple sampling positions.

5.2 Acoustic Body

For acoustic guitars, most of their sound comes from the vibration of the strings being transferred through the bridge of the guitar into the body, which would reverberate and amplify the sound. Therefore, we can sample near the bridge, and apply a convolution with an impulse response sample which carries the information of the body reverberation and microphone or piezo pickup signal transformation to simulate the acoustic body and use of microphones or piezo pickups.

6. VIRTUAL INSTRUMENT IMPLEMENTATION

To produce a practical virtual instrument prototype, we develop our program with JUCE in C++, which is a library that handles some parts of audio software implementation like audio processing, MIDI, VST interface and GUI.

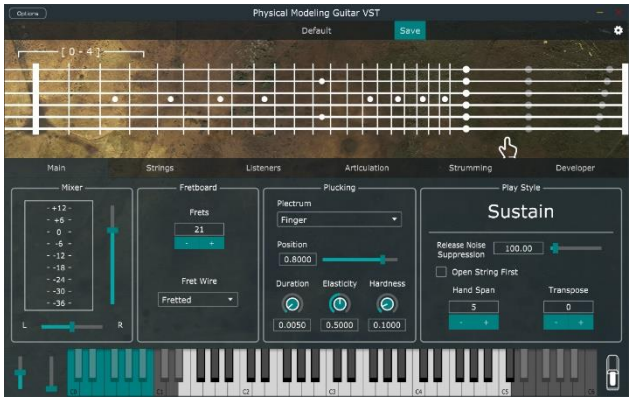

Figure 4. A screenshot of the prototype program.

To simulate a full guitar, we have multiple copies of the string model running at once (six for a typical guitar), each tuned to a different note. When a MIDI note is played, an appropriate string is chosen to play the note while minimizing the hand movement, to mimic the behavior of a guitarist. We also provide a strumming mode so the fingering when playing multiple strings at once (or with small time offsets) can be controlled more easily.

For performing and controlling different articulations, we define a MIDI map for the ease of MIDI programming and live performance. We also provide a set of key-switches to switch between different articulations modes.

To give more fine-grained controls over the guitar articulations, the note-on and note-off velocities of the key-switches and normal notes also affect the way a note is played. For example, the note-on and note-off velocities under the default articulation mode determine the plucking intensity and release speed respectively.

7. PERFORMANCE

In order to provide stable simulation and good audio quality, the simulation must run at least 44.1k cycles per second with 100 string nodes. Further increase of the temporal and spatial resolution would still lead to audible quality differences. With six strings and even multiple instances of the program, the program may not be practical for an average consumer-level computer to run smoothly. Therefore, optimization is necessary.

We make use of many techniques to improve the performance such as multithreading for different strings and redundant code reduction. With these optimization techniques, the program has been significantly accelerated.

7.1 Single Instruction Multiple Data (SIMD)

Another important optimization technique we use is SIMD, which is a technique to operate multiple data as vectors in a single instruction so that the whole process can be accelerated. It means that we can utilize SIMD to compute multiple string nodes at the same time.

However, it cannot be achieved easily in the actual case because of the essence of the finite difference method. Originally, we store the nodes sequentially in an array and then group the nodes into different vectors in that order. However, each node needs to access its adjacent nodes during computation, some of which would be in the same vector and some would not, such inconsistent accessing cannot be done efficiently with SIMD operations.

Therefore, we use an alternative string node ordering method as shown in Fig. 5. With this method, every node can simply access the same element location in the adjacent vectors. An extra vector needs to be constructed for the first vector to access its previous vector in every simulation cycle. It is similar for the last vector. However, the number of extra vectors is fixed. Hence, this cost is not significant when the spatial resolution is high enough.

v_0:	[0	8	16	24]	v_4:	[4	12	20	28]
v_1:	[1	9	17	25]	v_5:	[5	13	21	29]
v_2:	[2	10	18	26]	v_6:	[6	14	22	30]
v_3:	[3	11	19	27]	v_7:	[7	15	23	31]

Figure 5. A list of vectors with size four, using the alternative ordering method. Here the numbers in each vector v_i represent the index of the actual string nodes.

The general rule for this ordering method is defined by:

$$f(i) = N(i \bmod Size) + \left\lfloor \frac{i}{Size} \right\rfloor \quad (11)$$

$$g(j) = f^{-1}(j) = Size(j \bmod N) + \left\lfloor \frac{j}{N} \right\rfloor \quad (12)$$

where f is the function for mapping the index i (starting at zero) in the array to the index (starting at zero) of the actual node, g is the function for doing the inversed mapping, *Size* is the vector size, and N is the number of vectors.

7.2 Performance After Optimization

A performance test was conducted with an AMD Ryzen 5 1600 six cores 12 threads CPU running at 3.6 GHz, with 16 GB DDR4 RAM running at 3000 Mhz. Four versions including the program without optimization, with optimization (except for SIMD), with SSE and with AVX2 are tested. For the optimized versions, we also tested their performance with single threading and multithreading.

During the test, the simulation cycle was ran one million times at 120 spatial resolution with six strings, and the total computation time was measured. We ran the test three times to obtain the average time rounded to the nearest millisecond. As shown in Fig. 6. the performance improves significantly after optimization and can be used as a practical real-time music production program.

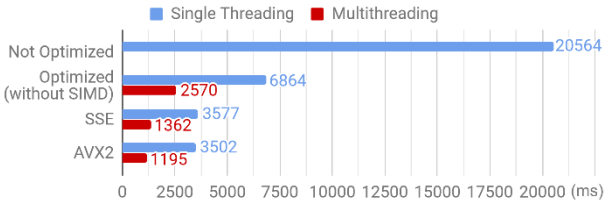

Figure 6. The result of the performance test, showing the performance difference before and after different kinds of optimization.

8. CONCLUSIONS

We have developed a guitar simulation with a physical string model, with an additional collision system that can simulate common guitar articulations. Convolution with impulse response samples is utilized to improve the realism of the simulated sound. Moreover, with optimization, the prototype program of our model can run reasonably well on desktop computers.

There is still room for improvement, such as the presence of longitudinal wave and sympathetic vibration, but we consider our simulation sufficient for producing natural and convincing simulated sound for music production.

9. REFERENCES

[1] C. McKay, "A survey of physical modelling techniques for synthesizing the classical guitar," 2003.

[2] H. P. Langtangen and S. Linge, Finite Difference Computing with PDEs A Modern Software Approach. Cham: Springer International Publishing, 2018.

[3] M. Shuppius, "Physical modelling of guitar strings," presented at Audio Developer Conference 2017, London. [Online]. Available: https://www.youtube.com/watch?v=sxt5rxF_PdI. [Accessed: 16-Dec-2019].

[4] J. Harriman, "Filtering Techniques for Piezoelectric Transducers." [Online]. Available: https://ccrma.stanford.edu/~jiffer8/420/project.html. [Accessed: 16-Dec-2019].