Band Overdrive: A Multi-Instrument Virtual Reality Music Rhythm Game

Junjie Wang* Shuqi Liao † Hao Wang‡ Christos Mousas§

Department of Computer Graphics Technology, Purdue University, West Lafayette, Indiana 47907, USA

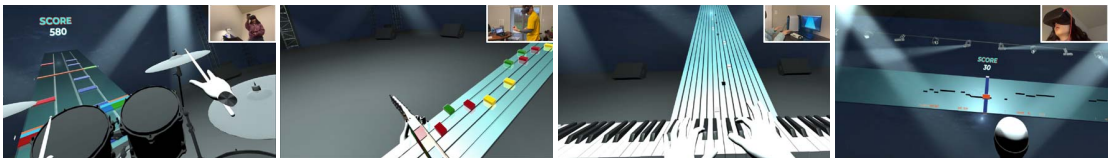

Figure 1: Four users in different locations play music together using the developed networked music rhythm game.

Abstract

Music rhythm games have a tremendous audience in the modern game industry. However, most of these games have been deployed for mobile devices, desktop computers, or game consoles. Although users can still experience the excitement of playing music by tapping on-screen buttons or using controllers, the user experience still has more room for improvement. We therefore developed a network-based multiuser virtual reality music rhythm game in which users can choose their preferred instrument or sing as a band while also collecting points based on their performance.

Index Terms: Computer Graphics—Graphics Systems and Interfaces—Virtual Reality; Applied Computing—Arts and Humanities—Sound and Music Computing

1 Introduction

Nowadays, music rhythm computer games are no longer limited to gameplay that challenges the user's sense of rhythm and fulfills the purpose of entertainment [1]. These games have shifted their gameplay objectives closer to the realistic simulation of specific musical instruments' detailed interaction and performance. For example, the *Rock Band*[1] and *Guitar Hero*[2] series are successful examples of music rhythm games that significantly influenced both the game and music industries [2, 3].

Virtual reality (VR) can provide a more diverse gaming experience and a more realistic simulation of music rhythm games due to the ability of users to use hand motions and gestures to interact with different instruments. However, only a few VR rhythm games use the full potential of VR to provide immersive gameplay and more advanced user interaction. Thus, we present *Band Overdrive* (our implementation can be found on our project's GitHub[3] page). This network-based multiuser music rhythm game provides ensembled rock band instruments that users can experience while being part of a band and playing their favorite songs on one of the provided instruments. *Band Overdrive* (see Figure 1) provides user-friendly interactions in VR and a more realistic simulation and experience of musical instruments.

*e-mail: wang4982@purdue.edu
†e-mail: liao201@purdue.edu
‡e-mail: wang5329@purdue.edu
§e-mail: cmousas@purdue.edu

[1]https://www.rockband4.com
[2]https://www.guitarhero.com
[3]https://github.com/BandOverdrive

2 Related Games

Rock Band and *Guitar Hero* are two of the most popular rhythm games on consoles and desktop computers (client-side) that provide game-based simulations of different instruments. However, both games require additional devices (specialized controllers), which could be considered inconvenient for most users and increase the cost of acquiring such games. *Unplugged*[4] is a VR guitar simulation rhythm game for Oculus Quest. In this game, using hand tracking, users are asked to match their fingers to the provided chords and strum the strings at the right time. *Paradiddle*,[5] a VR drum simulation application from the Side Quest community, provides users with the ability to assemble their virtual drum set and then use the VR controllers to play the drums. *Band Overdrive* considers the advances of these music rhythms games and instrument simulations and provides multiuser VR gameplay while also offering a better gaming experience.

3 Virtual Instruments

We implemented four virtual instruments: a drum set, guitar, bass guitar, and keyboard. We also provide the ability for users to sing karaoke-style.

3.1 Drum Set

We implemented a virtual drum set, which the users can play via VR controllers and hand movement. Each virtual hand holds a drumstick. When a collision between a drumstick and the different drum parts (drums or cymbals) is detected through a collision trigger function, the system plays the corresponding sound. During gameplay, a drum tablature appears in front of the drums to instruct the user which drum part to hit with the drumstick. Figure 2 shows the developed drum set scene.

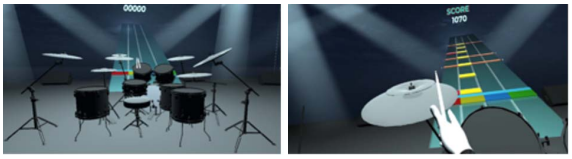

Figure 2: Drum set scene (left) and drum gameplay from the user's point of view.

3.2 Guitar and Bass Guitar

Our guitar and bass guitar playing method provides natural interaction to users through hand tracking. We implemented hand tracking using the Oculus hand tracking[6] functionality, and based on the

[4]http://unplugged-vr.com
[5]https://paradiddleapp.com
[6]https://developer.oculus.com/documentation/unity/unity-handtracking

978-1-6654-8402-2/22/$31.00 ©2022 IEEE
DOI 10.1109/VRW55335.2022.00304

hand tracking data, we fix the relative position between the hand and guitar (or bass guitar). Moreover, by using the hand tracking functionality, we can retrieve the finger bones' ID, which is used to detect collision between fingers and buttons to ensure the users are pressing the correct button on the guitar. Lastly, the colorful rhythm notes on the tablature are also present in the virtual scene to instruct the users which buttons to press at each time step based on the selected music. In Figure 3, we illustrate examples of our guitar playing scene.

Figure 3: Guitar playing scene from the user's point of view.

3.3 Keyboard

To enable users to play such an instrument, we used the hand tracking functionality of Oculus Quest. Using their fingers, users can press the different buttons on the keyboard. Button pressing is achieved using a collision trigger functionality. It should be noted that, in our application, we provide only a two-octave range. Also, to simplify keyboard playing, we provide tablature-style instructions to users. In Figure 4, we illustrate the developed scene for the keyboard.

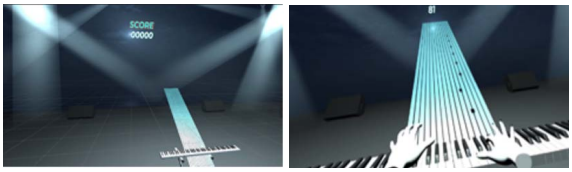

Figure 4: Developed keyboard scene (left) and keyboard playing from the user's point of view (right).

3.4 Vocals

We also implemented vocals to allow users to sing karaoke-style. The implementation of vocals has a different mechanism compared with the previously mentioned instruments. To implement vocals, we used the Human Voice Pitch Detection[7] plugin to get the pitches from human voice input and for voice detection analysis. It should be noted that, because voice recording is considered personal data, we ask users to provide recording permission before the game starts. In Figure 5, we present the scene that was used for vocals.

Figure 5: Vocal playing scene.

4 Rhythm Gameplay

The gameplay development of our method consists of two parts: 1) MIDI file processing (including file reading and analysis) and 2) building the rhythm gameplay engine.

[7]https://assetstore.unity.com/packages/tools/audio/human-voice-pitch-detector-109019

MIDI File Reading and Analysis. We used DryWetMIDI[8] for MIDI file reading. After importing a MIDI song to our system, DryWetMIDI reads the file and extracts each instrument's audio dataset. Then, based on the user's selected difficulty, the musical notes are subdivided into four levels: "Easy," "Medium," "Hard," and "Expert." Lastly, we get a list of notes of the current corresponding instrument and difficulty, in which a note consists of its time and lane position. To perform the correct rhythm game, we split the note list by lane.

Gameplay Data Structure. Each gameplay scene (i.e., drum, guitar, bass guitar, keyboard, and vocals) needs to inherit a "Track," "Lane," and "Note" class to generate the rhythm gameplay. Specifically, the "Track" script is the control center for the rhythm gameplay. This script includes the song and score managements. In song management, "Track" works for audio source loading and analysis. In score management, it works for updating the score based on the hitting rhythm strike accuracy. "Lane" is used to generate the series of note prefabs based on the input note list from "Track" and to ensure time alignment between the notes and the audio. Each lane on the tablature represents one type of musical note in a track, and only one track will be defined in one instrument. "Note" is responsible for rendering the selected note (sound). It should be noted that the "Track," "Lane," and "Note" classes can have corresponding child classes based on the instrument type, and each gameplay scene features its own program processing based on its respective attributes.

5 Multiuser Mode

We used the Photon Unity Networking 2 (PUN2)[9] to implement the multiuser mode of our game. In the multiuser mode, each user can choose different rooms in the lobby and then enter team play. After the assigned users in the same room select their role (i.e., instrument), the master client of the room can get the control canvas to set up the common playing song. After choosing a song and its difficulty level, the master client can start the game for all the users in that room. Then, each user can play the instrument in the team based on their selected roles.

6 Conclusion and Future Work

We developed a multiuser VR music rhythm game for Oculus Quest in the Unity game engine. Our game allows the user to play one of four implemented instruments (i.e., drum, guitar, bass guitar, and keyboard), or to sing (vocals). We automated the rhythm gameplay process by implementing a method to read and analyze the imported music files.

We will concentrate on improving the VR gaming experience in our future work. Among other goals, we want to improve the realism of the developed scenes, create more realistic musical instrument gameplay, revise the synchronization of the multiuser mode, and create a more user-friendly interface. Lastly, we want to conduct user studies to understand how users experience such a VR game.

References

[1] D. Arsenault. Guitar hero:" not like playing guitar at all"? *Loading*, 2(2), 2008.

[2] K. Miller. Schizophonic performance: Guitar hero, rock band, and virtual virtuosity. *Journal of the society for American Music*, 3(4):395–429, 2009.

[3] D. Roesner. The guitar hero's performance. *Contemporary Theatre Review*, 21(3):276–285, 2011.

[8]https://melanchall.github.io/drywetmidi/index.html

[9]https://doc.photonengine.com/en-us/realtime/current/getting-started/realtime-intro