Virtual Music Space

Bader Tayeb Evan Zien Luan Pham Will Ceriale

Abstract

The Virtual Music Space provides a new way to experience music that involves the user with more than their sense of sound. With the Oculus Rift, our virtual reality application will allow someone to visualize as well as interact with any music of their choice. Currently on the market, there are no music visualizers that take advantage of the Oculus Touch, and they do not allow users to really immerse themselves in the music through interaction. Our music visualizer solves this problem and allows the user to interact fully with their music with the ability to move around objects and play with sound locations. This is useful to someone because people get to really immerse and enjoy their music to a new level. The Virtual Music Space solves the problem all current music visualizers face on the market, and it allows people to be more engaged with their music.

1 - Introduction

Music visualizers have been existed for a while and are not abundant in today's world. Unfortunately, most of these music visualizers are only 2D and do not really immerse the user in the music. The rise of virtual reality creates the opportunity for music visualizers to exist 3D space. Currently on the market, most music visualizers do not give the user the opportunity to interact or move within their environment.

Our goal is to explore the idea of connecting the interaction between music, objects, and the user. We used the virtual reality headset, the Oculus Rift, and its controllers, the Oculus Touch, to carry out this goal.

2 - Solution

Our solution to this issue is Virtual Music Space. A 3D music visualizer where the user is able to freely float in space and interact with any of the objects and even creating new ones. Virtual music space application is developed using Unity 2017.2.0. All scripts are written using C#.

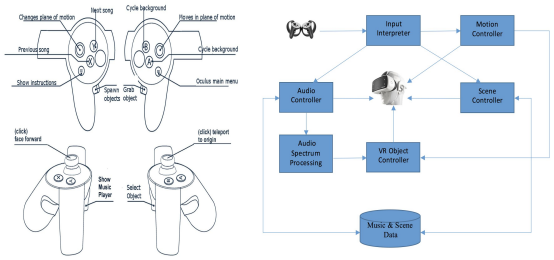

Figure 1: Left: Controller Input Mapping. Right: Interaction Between Major Components.

2.1 - Design

Virtual Music Space contains several interacting major components (Figure 1 Right). Input signal from wireless controller is interpreted and attached to corresponding event handlings. Motion type signal is sent to Motion Controller to interpret user actions. Audio control type signal is sent to Audio Controller to give access to music resources and performs user designated tasks on music playlist. Audio Spectrum Processing converts input signal to frequency domain, which in turn allows VR Object Controller to set object movements. Scene control type signal is sent to Scene Controller to give access/control to images resources.

2.2 - Component & Implementation

2.2.1 - Input Controller:

Describe: Wireless controller allows user to grab/move objects, trigger menu buttons and control data resources. It is one of the main input signals in our VR application.

Implementation: In order to maximize user experience, especially in VR environment setting. It is important for user to visual their hands and movements in real-time. We tackle this problem using Oculus Avatar SDK to render both hand controllers at run-time.

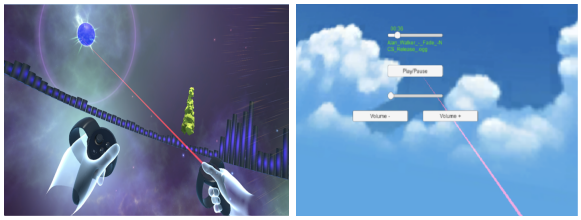

Figure 2: Left: Grab/Rotate/Move object using pointer. Right: Predefined collider detects a ray hit event.

2.2.2 - Input Interpreter:

Describe: Input interpreter is the mapping between input keys and event handlings (Figure 1 Left).

Implementation: Unity has a very nice feature that allows user to map majority input keys to event handlings. Within each event handling, we can define scripts to perform designated tasks. The only exceptions are thumbstick, index and hand trigger. Since we do not have a direct map on those buttons, we utilize Oculus Avatar SDK from Oculus to implement a separate set of event handlings.

All interactions in our VR application are simulated using laser pointer. Direction of laser pointer is obtained from forward vector of right hand controller. For each object we define a collider ; we indicate a ray hit on targeted object by changing color of laser pointer to red (Figure 2 Right). Clicking action is simulated by the detection of the collider hit by the forward ray. We incorporate this idea to define Play/Pause, Volume+/- (Figure 2 Right). On the other, Grabbing action is triggered during a ray hit event while user holds the right hand trigger button (Figure 2 Left). Moving object is a process of determine the distance between object and controller. The result distance and new forward vector are combined to set new location of the object. In a similar manner, rotating/pulling/pushing action is computed using the difference between current position/angle and previous position/angle. Result difference is scaled in the direction forward vector in the case of pulling/pushing (we update these parameters every single frame).

2.2.3 - Motion Controller:

Describe: Motion Controller is the main interface to control movement in VR environment. The user uses the two thumbsticks to move in space. We used one thumbstick to move in a 2D plane and the other thumbstick to change the rotation of the 2D plane in 3D space, giving the user the freedom to move anywhere inside the space. Movement is very accessible in this application.

Implementation: Movement is implemented using a simple event handling to adjust position as well as OVR camera facing direction. We determine the user positions on the thumbstick trigger and convert it to the corresponding moving distance and rotation angle along that axis (we scale both distance and angle by constants).

2.2.4 - Audio Controller:

Describe: Audio Controller is the main interface to control music playlist. Music can be added in itself by dragging in music to a folder in the application. Music songs can be cycled with a button. In addition, user can experiences spatial audio effects as they move around the music sources.

Implementation: Music files are extracted from data resources and stored as input array. Index on music playlist is updated as user triggers a designated keypress event or when the playlist stops playing (end of current song). We also integrate an automatic forward feature using a separate flag variable to keep track of the playing state of music playlist (this state variable is independent from the playing state defined by user).

Each audio source has two major attributes to support spatial audio effects: volume, radius. The behavior of these two attribute are integrated inside Unity for VR. Volume scales linearly to the distance from music source; whereas, radius defines the effective range for audio source (sounds of the sun and moon are active within a predefined radius). The Tree of Life (our music source) is the music of the music space and you can hear it differently depending on its location with respect to you.

2.2.5 - Scene Controller:

Describe: Scene Controller is the main interface to control background images.

Implementation: We use very similar approach to Audio Controller to keep track of current index on input array of background images.

2.2.6 - Audio Spectrum Processing:

Describe: Audio Spectrum Processing reads music input from Audio Controller, performs FFT to transform time domain to frequency domain.

Implementation: Data is obtained by sampling input audio source and stored as an array (we define a fixed size array of 512 to store sample data). FFT algorithm is performed on sample data to obtain corresponding frequency domain. Having the frequency domain, we can determine human hearing frequency (8 bands spanning from 20Hz - 20KHZ) by averaging the frequency values within each interval. To allow objects to bounce to the beat, we keep track of highest frequency for each frequency band on each running frame. The radio between current frequency and highest frequency tells how frequency expands or contracts at a given time frame. Summing all ratio values of all 8 bands together and update its highest value at runtime, the ratio should give us how object react to the music beat.

Figure 3: Left: Bars represent frequency Spectrum (they are scaled by a constant). Right: Object size is scaled to the amplitude of frequency.

2.2.7 - VR Objects Controller:

Describe: VR Object Controller is the main interface to read spectrum data from Audio Spectrum Processing unit and convert these data to corresponding color coding and object movements.

Implementation: For each object we define a function to obtain updated frequency and amplitude from Audio Spectrum Processing unit. Objects in the shape of a cylinder, pill, sphere, and cube can be created and they alter size based on computed amplitude on each frame (Figure 3 Right). There are also music bars following the spectrum frequency (Figure 3 Left). We also try to create lighting & color effect by applying computed amplitude, frequency to adjust the color of objects. However, this approach causes a significant performance degradation (due to re-rendering multiple objects every single frame); thus, we decide to omit this feature during the demo.

3 - Related Work

There are two applications in the Oculus Store that offer immersive music visualisation. These are "Virtual Music", and "Audio Visualizer". Virtual Music (Figure 4 Right) is a music visualizer that had good static designs that reacted with music but lacked a clear interaction with the user. The user could not pick the music being played, nor could he/she move or interact with the space. Similarly, Audio Visualizer (Figure 4 Left) had the same problems with lack of interaction. Audio Visualizer included more moving visualizations, but it still could not fully engage the user with interactions with the music. Audio Visualizer only had visuals on a 2-D plane and was only a visual for the user. Both of these apps take advantage of the Oculus Rift and place the user in a fully immersive space where objects react to music. A key difference between our project and the ones that existed previously is the level of immersion. In our application, one is able to move objects, spawn objects and completely move around the world, where in previous application movement was limited, as well as object interaction.

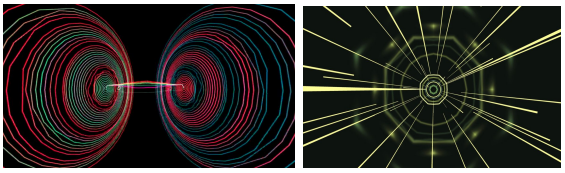

Figure 4: Left: Entire Visualization Space on Audio Visualizer. Right: Entire Visualization Space on Virtual Music

4 - Conclusion

We were able to create a fully immersive virtual reality experience through the Oculus Rift and the Unity Engine that is not in existence on the Oculus Store. Currently, music visualizers on the market only have visualizations that follow the pitch and frequency of the music. Users are only able to view these objects.

While the object look impressive, the users still cannot make any interactions with them. Our application, Virtual Music Space, alleviates this void and gives users full interaction with the objects. Despite accomplishing a lot this past quarter, this project could still be expanded in the future. For example the designs on creating objects can be much better and give more interactions such as deleting objects or bumping objects. This project can even expand to letting people experience live music through an Oculus because of the interactions. Virtual Music Space accomplishes our goal of creating a more immersive music visualizer, and it has potential to impact a larger scope if carried out further.

5 - Acknowledgements

We would like to thank Peer Play for amazing Audio Visualization tutorial series on YouTube (https://www.youtube.com/channel/UCBkub2TsbCFIfdhuxRr2Lrw). These tutorials are super helpful as we get to learn more about Unity features along the way. We also would like to thank professor Bruce HemingWay and TA Tony Tung for giving us helpful feedbacks, which inspire us to add more amazing features to our final product.