Indeterminate Sample Sequencing in Virtual Reality

Chase Mitchusson
Louisiana State University
340 E Parker Blvd
Baton Rouge, Louisiana 70803
cmitc79@lsu.edu

ABSTRACT

The purpose of this project is to develop an interface for writing and performing music using sequencers in virtual reality (VR). The VR sequencer deals with chance-based operations to select audio clips for playback and spatial orientation-based rhythm and melody generation, while incorporating three-dimensional (3-D) objects as omnidirectional playheads. Spheres which grow from a variable minimum size to a variable maximum size at a variable speed, constantly looping, represent the passage of time in this VR sequencer. The 3-D assets which represent samples are actually sample containers that come in six common dice shapes. As the dice come into contact with a sphere, their samples are triggered to play. This behavior mimics digital audio workstation (DAW) playheads reading MIDI left-to-right in popular professional and consumer software sequencers. To incorporate height into VR music making, the VR sequencer is capable of generating terrain at the press of a button. Each terrain will gradually change, creating the possibility for the dice to roll on their own. Audio effects are built in to each scene and mapped to terrain parameters, creating another opportunity for chance operations in the music making process. The chance-based sample selection, spatial orientation-defined rhythms, and variable terrain mapped to audio effects lead to indeterminacy in performance and replication of a single piece of music. This project aims to give the gaming community access to experimental music making by means of consumer virtual reality hardware.

Author Keywords

NIME, proceedings, virtual reality, sequencer, indeterminate, sampler

CCS Concepts

•**Applied computing** → **Sound and music computing;** *Media arts;* Performing arts;

1. INTRODUCTION

The VR Sequencer is made for composition and performance, and relies on sample playback as its audio source. Making music with the VR sequencer is a playful experience where the user can interact with a large space to compose or perform. A user enters a VR world where they are standing on a 250 square-meter plane. To begin composing or performing, they may enter a menu to select an item to spawn on the stage. If they select a die, it will appear and they can pick it up and throw it wherever they would like. There will not be any audio, yet, however. They must then spawn a time-and-space sphere. Once the sphere is spawned, it will grow in size up to 150 meters in diameter. When it contacts the previously thrown die, it will trigger the audio clip selected by the die. Finally, there is music! How about spawning another three or four dice and throwing them about? A musical pattern emerges from the looping growth of the sphere. How about complementing the pattern with a new one by adding another sphere to the stage? Now there are two musical patterns of the same material. With a few adjustments to one of the spheres, it will change speed and trigger a different set of samples to play. Now there are two patterns with two musical ideas from the same dice!

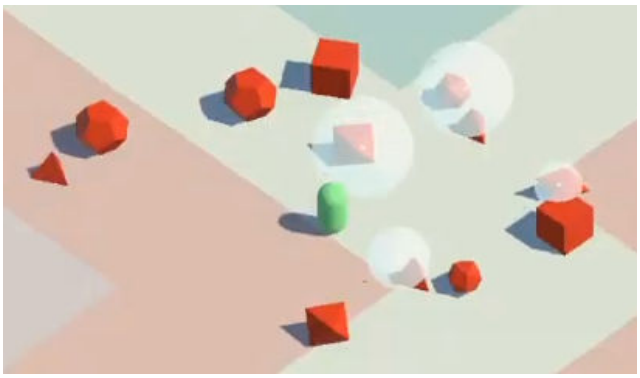

Figure 1: Third-person view of the VR Sequencer with four time spheres and twelve dice.

The VR sequencer uses multiple looping processes and multiple audio banks to create music based on spatial positions of dice and spheres as shown in Figure 1. It is being developed using Unity version 2019.1. All scripting of audio processes, asset generation, and user interfaces are done with C#. The VR kit for which the sequencer is being developed is the HTC Vive on Windows 10.

2. PRIOR WORK

Many musical VR projects choose to use synthesis, rather than audio clips, which impart the feeling of playing an instrument and allow a performer's virtuosity to strongly impact the resulting sounds. Rob Hamilton's *Carillon* does this by incorporating additive synthesis via Pure Data combined with a performer's skill in hand-eye coordination and musical timing to play bell instruments [4]. VR instruments

Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). Copyright remains with the author(s).

NIME'20, July 21-25, 2020, Royal Birmingham Conservatoire, Birmingham City University, Birmingham, United Kingdom.

developed by Mäki-Patola, Laitinen, Kanerva, and Takala use custom gloves and sensors to mimic instruments like xylophones, guitars, and membranes while utilizing Karplus-Strong physical modeling techniques to produce audio [6]. Given that their goal was to make virtual versions of physical instruments, the performers generally have some level of familiarity with how to play the instruments, and their virtuosity may depend on how the virtual version differs from the physical.

The sample playback design of the VR sequencer removes synthesis from the project in favor of indeterminate musical choices, à la John Cage's *Music of Changes* [5]. Removing synthesis does not define indeterminacy, as proven with the 3-D reactive widgets developed by Berthaut, Desainte-Catherine, and Hachet which use audio clips triggered by performer gestures [1], but the method by which clips are selected can define the level of indeterminacy. Dice in the VR sequencer have no labels and are monochromatic. A user is completely unable to tell what the selected sample will be until it is heard. The looping mechanism in the VR sequencer plays at a regular interval, generating a form of control over random input [5].

There is a theme of connecting various digital media to design these virtual instruments, which the VR sequencer does not embrace. Use of leap motion devices [7, 3, 4], tracking gloves [6, 2], and software communicating from outside the 3-D engine [4] lends itself to high levels of experimentation and customization, but is not particularly well suited for commercial use or access. The VR sequencer is designed to be enjoyed by owners of consumer-grade virtual reality hardware. This choice opens access to experimental music making to the gaming community, who may not be familiar with writing and performing experimental music. Getting a copy of the VR sequencer should be as easy as downloading it from the Steam store and playing with the hardware and software VR gamers have at home. Custom hardware and extraneous software inhibit the ability to share these VR projects with non-experimental musicians.

3. THREE-DIMENSIONAL ASSETS

Sequences are created by interacting with two types of 3-D assets, dice and spheres. To make music, a user must roll a die and have it come in contact with a time sphere. The dice decide what audio clip to play back and the time sphere decides when to play it back.

3.1 Dice

The 3-D assets which represent audio clips are actually audio folder containers that come in six shapes. Five of the shapes are polyhedral with 4, 6, 8, 12, and 20 sides, while one is a 10-sided trapezohedron. These are all common dice shapes. They were chosen because the sequencer has indeterminate qualities, and dice work well in chance-based operations. The dice are designed to be grabbed by a user when the user contacts the Vive controller with a die and pulls the Vive controller trigger to pick up the object. Once held, the dice can be thrown across the scene to land on an indeterminate face. The face of a die that is touching the ground in the VR sequencer determines which sample is triggered by the time sphere. Music is composed with the VR sequencer by rolling dice. The chance-based sample selection and spatial-location-defined rhythms lead to indeterminacy in performance and replication of a single piece of music.

3.1.1 Audio Clip Assignment

The Unity project's Assets folder contains all assets in the VR sequencer. Within the Assets folder is an Audio sub-folder which contains its own sub-folders full of audio clips. The audio clips from each folder are assigned to a face of each die using Unity Web Request. Unity Web Request reads a file path and provides the file to Unity as needed. In this case, Unity Web Request reads the Audio folder to find the number of sub-folders, then reads each sub-folder to find the WAV files at the end of the path. It assigns the WAV files of every sub-folder to each new instance of a die until it reaches the number of faces on the given die. So a four-sided die (D4) will contain the first four WAV files of each Audio sub-folder, and a ten-sided die (D10) will contain the first ten WAV files of each Audio sub-folder, and so on.

The decision to give them audio clips from every folder is due to the time spheres doing the actual playback triggering based on a layer number. If there are five Audio sub-folders, then there are five layers for the time sphere to choose. Time spheres on layer 3 only trigger samples from the third Audio sub-folder, and time spheres on layer 5 only trigger samples from sub-folder five. That way, a single die can have $5 \times n$-number of sides worth of samples accessible by users.

This method shows favoritism toward the first audio clips in a folder, so by altering the C# scripts of each die, one can arrange a D4 to always play the first four clips and a D10 to play the last ten clips. This behavior can be set to different layers as well. A D4 on layer 1 will read the first four clips, but will read clips five through eight on layer 2. Leaving the programming modular in this manner creates opportunities to avoid tedious repetition of audio clips.

3.2 Time-And-Space Spheres

Spheres which grow from a variable minimum size to a variable maximum size at a variable speed, constantly looping, represent the passage of time in this VR sequencer. As 3-D assets come in contact with a sphere, their samples are triggered to play. This behavior mimics digital audio workstation (DAW) playheads reading MIDI left-to-right in popular professional and consumer software sequencers.

Each sphere has a visible, grabbable portion called the space sphere, and an intangible portion which loops and interacts with the audio objects called the time sphere. The space sphere is movable when grabbed, but is unaffected by gravity. Therefore, users can grab a space sphere, move it to a new location, and let the sphere hang in the air where they release it. Time spheres are attached to the space sphere's location transform, wherever the space sphere goes, the time sphere will follow. However, the user cannot collide with or grab a time sphere, they must interact with the space sphere. The space sphere is like a glass orb and the time sphere is like light pulsing out of the orb.

The two spheres together are referred to as time-and-space spheres. To spawn a time-and-space sphere, users click the menu button on the Vive controller, which brings up a menu where they can click a button to instantiate a new time-and-space sphere. The sphere will spawn in front of the user, and from there users can interact with it. To interact with the sphere, a user must contact the space sphere with the Vive controller and pull the Vive controller trigger to grab it. Once grabbed, the space sphere can be placed anywhere in the scene. While holding the space sphere, the user can click the menu button to bring up the time sphere's editable parameters, shown in Figure 2. These parameters are layer, minimum size, maximum size, speed, reverse, and delete.

3.2.1 Layers

Layer represents a folder containing audio files. There are several folders with various samples in each one that are

assigned to the dice. Choosing a layer decides which audio folder to read from when interacting with the dice. For instance, layer 1 relates to folder 1 which may contain guitar samples, and layer 2 relates to folder 2 which may contain drum samples. The layer chosen for the time sphere determines what sample plays when it contacts the dice. This is done so that multiple time-and-space spheres in a scene can add variety to the sample playback.

3.2.2 *Minimum and Maximum Size*

Minimum size determines the smallest size the time sphere can be and maximum size determines the largest size the time sphere can be. The default minimum size is a diameter of 0 meters, and the maximum default is a diameter of 150 meters. Changing the size alters which dice come in contact with the time sphere, therefore altering the melody of the music. A time sphere with a minimum of 80 meters and a maximum of 90 meters only comes in contact with a narrow portion of objects, while a minimum of 0 and a maximum of 125 will contact nearly every object in the scene. Users can take advantage of size to be selective of which dice they want to trigger.

3.2.3 *Speed*

Speed refers to the time it takes for the time sphere to go from minimum size to maximum size. Speed is listed as integers on a slider, the default being 1. The higher the number, the faster the time sphere moves. Users can change the speed parameter when they want to affect the tempo of a given sphere. There is a hidden unit as well called duration which can change the scale of speed units. A speed of 1 and duration of 1 means one second of time, but a speed of one and a duration of 0.5 means two seconds of time.

3.2.4 *Reverse*

Reverse is a check box that determines which direction the time sphere moves. When unchecked, the time sphere grows from minimum size to maximum size, but when reverse is checked it shrinks from maximum size to minimum size. This feature reverses the playback order of the dice it contacts, which adds variety to the music. It is the equivalent of reading a MIDI file in retrograde motion in a traditional DAW.

3.2.5 *Delete*

The delete button does exactly as one may suspect, it deletes the currently selected time-and-space sphere. When composing or performing with the VR sequencer, deleting spheres breaks up the monotony in musical patterns and loops that have been playing too long.

4. MUSICAL INTERACTION

Users need a way to quickly make large musical changes. To incorporate a mass music mix-up, a terrain generator has been developed which jumbles the dice at the click of a button. When users press the grip button on the controller, a perlin noise map is generated and applied to the terrain mesh. To smooth out the transition from an old terrain to a new one, the noise and height values interpolate from old to new over a span of a random time between four and twenty seconds. The changes can be extremely subtle, barely changing over long periods of time and doing little to jumble the dice, or the changes can be drastic, changing quickly over a few seconds, sending dice flying into the air, shown in Figure 3.

Connected to the terrain changes are audio effects changes. There are three different stages with three different audio

Figure 2: First-person view of a time-sphere having its parameters edited.

effects chains connected to the terrain. In Unity, audio effects can be made into presets called snapshots, and snapshots can be called at run-time. When a user clicks the grip button to change the terrain, a different snapshot is called and the values interpolate to the new preset over the course of the terrain change. This adds another level of indeterminacy to the music. The effects, combined with terrain values and dice being re-rolled create what feels like a new section in a piece of music, and can have potent impact on the form of the music.

5. EVALUATION

Standard interactions such as spawning dice and time-and-space spheres are working as designed. Grabbing dice is intuitive, but the large size and shape of the dice, about two meters tall, means throwing them sidearm like a baseball pitcher can allow them to roll smoother and further than using an overhand throw. The playback of audio based on time sphere values is also working as designed. The biggest sphere related hiccup is the UI menu. After adjusting a time sphere's values, users must "let go" of the sphere before walking away, or risk the wrong menu opening next time they want to edit or spawn something. Also, It is not initially easy for people to click menu buttons due to sensitivity of the VR controller. When users click the controller's trigger, it causes their hand to move and possibly mis-click. Due to this, spawning items is a challenging, yet achievable, hurdle for new users.

The most useful features of the current version are the terrain changes and the time sphere parameters. When starting a piece, the composer may want to add several time-spheres to play back one die, like in Figure 4. By editing the spheres carefully, they can achieve an interesting rhythmic and melodic idea. The layer parameter is an obvious first change to make the die play different samples, but changing the size and speed can enrich the music with rhythmic intricacies. Given that the time-spheres are constantly looping, the music can become tedious, especially when first starting a piece. This is because of the time it takes to spawn several dice and make any changes to the time-sphere. That tediousness is eliminated with the terrain changes. Once the composer deems the music as stale, they can click a button to change the terrain and create something new almost instantly. If what the terrain gen-

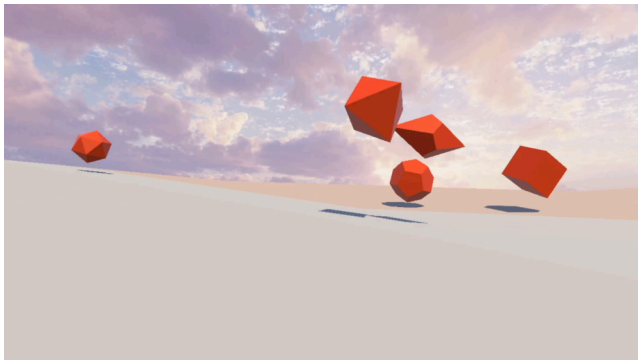

Figure 3: First-person view of the VR Sequencer terrain causing dice to bounce around.

erates is not worthy of the piece, again, the composer can click a button and move on to a new section. Since the audio effects are tied to the terrain changes, the outcomes of the changes can be really fascinating and worth wallowing in for a while. Each massive change can feel like a fresh start and lead to creative ideas on how to continue the piece.

6. FUTURE WORK

The size of the dice is novel, but for more comfortable interactions, they may need to be resized down to around 1 or 1.25 meters in height. That would hopefully lead to ease of rolling and throwing overhand, sidearm, and underhand. There needs to be a better way to program how and when the time-sphere edit menu pops up, since sometimes it pops up after it has been edited and is not still being held. It is also worth developing a better method of menu selection than the current raycasting and selecting items with a controller trigger. A UI that takes advantage of the controller's trackpad may be better suited for this application.

While a way to change everything at once has been incorporated, what is lacking is the ability for a performer to command specific mass interactions such as stopping four time-spheres from triggering audio at one time, or the ability to roll only all D4 at once. When it comes to performing, a constant loop can become grating to listeners, so allowing a performer to conduct the time spheres becomes a necessity. The spatial music production interface from Wozniewski, Settel, and Cooperstock tackles this issue well by allowing performers to dip the volume of source nodes, like using a sampler or mixing board to bring audio in and out [8]. This is a powerful tool that helps develop form and structure while writing music, and cleverly allows for mixing audio using 3-D space.

7. CONCLUSIONS

Music making in virtual reality should reach audiences at a consumer level. By stepping away from developing musical VR experiences with hardware and software outside the virtual reality headsets and controllers, people who lack access to special VR events, concerts, and conferences can now enjoy experimental music making in their home. Rather than focus on how acoustic instruments can be replicated in VR, the VR sequencer strives to relate more to musical samplers and sequencers. More than just looping audio clips, the VR sequencer adds indeterminacy in melody making and allows for multiple layers of loops at once to aid in creating interesting rhythmic ideas. Basing interactions around how objects relate to each other in space creates an unorthodox method for writing and performing musical ideas.

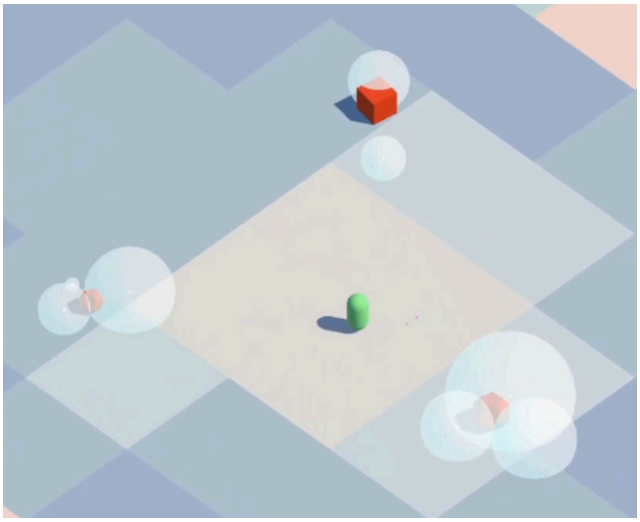

Figure 4: Third-person view of how multiple time-spheres can be used to play back a single die.

8. REFERENCES

[1] F. Berthaut, M. Desainte-Catherine, and M. Hachet. Interacting with 3d reactive widgets for musical performance. *Journal of New Music Research*, 40(3):253–263, 2011.

[2] M. Cabral, A. Montes, G. Roque, O. Belloc, M. Nagamura, R. R. Faria, F. Teubl, C. Kurashima, R. Lopes, and M. Zuffo. Crosscale: a 3d virtual musical instrument interface. In *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 199–200. IEEE, 2015.

[3] J. Fillwalk. Chromachord: A virtual musical instrument. In *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 201–202. IEEE, 2015.

[4] R. Hamilton and C. Platz. Gesture-based collaborative virtual reality performance in carillon. In *Proceedings of the international computer music conference 2016*, pages 337–340, 2016.

[5] M. G. Jensen. John cage, chance operations, and the chaos game: Cage and the" i ching". *The Musical Times*, 150(1907):97–102, 2009.

[6] T. Mäki-Patola, J. Laitinen, A. Kanerva, and T. Takala. Experiments with virtual reality instruments. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 11–16. National University of Singapore, 2005.

[7] A. G. Moore, M. J. Howell, A. W. Stiles, N. S. Herrera, and R. P. McMahan. Wedge: A musical interface for building and playing composition-appropriate immersive environments. In *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 205–206. IEEE, 2015.

[8] M. Wozniewski, Z. Settel, and J. R. Cooperstock. A spatial interface for audio and music production. In *of the 9th International Conference on Digital Audio Effects*, page 271, 2006.