Lisa Zahray 2/3/2017
6.UAP Final Report Supervisor: Eran Egozy

Exploration of Music Composition in Virtual Reality

Abstract

Virtual reality (VR) is a new technology with many aspects that have yet to be fully explored. This project continues previous work done over the summer to create and user-test two music composition sandboxes for the HTC Vive virtual reality system. The sandboxes were created using Unity and C#. The first sandbox is called the Grid Sequencer, in which the user places cubes into different grids to create looping audio. The second is called the Bottle Shaper, which focuses on the mechanic of reshaping bottle objects. The user can hit the bottles with a drumstick or place them in the path of a fan to produce different sounds. 16 participants playtested the sandboxes and completed a survey about their experience. The results show a few major patterns. Users preferred using in-world objects to manipulate song parameters as opposed to buttons on the controllers. Overall, users felt more immersed in the Grid Sequencer world than the Bottle Shaper world. Reasons for this include inconsistent physics in the Bottle Shaper, as well as a discrepancy between user expectations of an interaction versus what actually occurs in that interaction. Users were also more satisfied with the sound of the music they produced in the Grid Sequencer. This was largely due to a higher level of sound automation in the Grid Sequencer, versus the Bottle Sequencer's higher level of musical freedom. Users reported high enjoyment ratings for both games and offered many pieces of feedback, both specific to the sandboxes and regarding VR interactions in general. There are many possibilities for future work. This includes adding features to the sandboxes based off user feedback, and investigating entirely new music composition techniques in VR.

Background

While the idea of virtual reality has existed for many years, it wasn't until fairly recently that it became accessible to the public. Google Cardboard was one of the first widely popular headsets, released in 2014. It visually immerses the user in a virtual world, but does not allow the user to meaningfully interact with that world. The system's other main limitation is that it cannot detect head position. Spring 2016 saw the release of two high-end virtual reality systems: The Oculus Rift and the HTC Vive. The Oculus Rift was first released as a seated VR experience that allows the user to interact with their world through a standard XBox controller. An external camera tracks head position. The HTC Vive was released a week later as a full-room virtual reality experience. Two sensors mounted in the room allow for accurate motion tracking in a larger space. Additionally, the Vive originally came with a pair of motion-tracked controllers. The Oculus Rift recently released its own set of motion-tracked controllers.

Because this project focuses on music composition sandboxes, it is necessary to understand existing virtual reality experiences involving artistic creation, musical experience, and music composition. One example of an experience that focuses on artistic creation is Tilt

Brush [Google]. It is a creative sandbox where the user can paint in their 3-dimensional space. A game that does not involve artistic creation, but that focuses on musical experience, is Audioshield [Fitterer]. It is a rhythm game where the player uses shields to block incoming orbs in time to the music. It can calculate a track for any song the user selects. Harmonix Music VR combines artistic creation and musical experience [Harmonix]. It provides four different ways to experience pre-existing music. Some of these four modes are more passive, while others involve more interaction and creativity. A good example of a VR experience that focuses on music composition is SoundStage [Hard Light Labs].

SoundStage is a music composition sandbox that, in some ways, is similar to the type of experience this project focused on designing. It uses a modular approach to sound synthesis, allowing the user to attach components together and modify the layout of their own personal sound creation studio. Some of the main modules include a sequencer, a looper, a drum kit, and an oscillator. The sequencer is a 2-dimensional grid. The rows correspond to different sounds, and each column represents a different quantized moment in time. Notes play on squares that have been selected. The looper lets the user record a series of piano key presses and then loop them in the background. The drum kit is a live-performance tool, in which the user hits different drumheads with a pair of drumsticks. The oscillator module produces waves of different frequencies and shapes.

Project Goals

This project had two main goals. The first was to create two music composition sandboxes for virtual reality, focusing on different types of interaction and composition tools. The second was to playtest the sandboxes to investigate the effectiveness of these features in the context of virtual reality.

The existing VR experiences previously described each have aspects that work well, but they also seem to under-utilize some of the features VR enables. Google Tilt Brush provides a great visual creative experience, but does not allow the user to express themselves musically. AudioShield is a fun way to interact with music, but it under-utilizes the space. The user stands in place and always faces one direction. Harmonix Music VR creates a fun musical experience, but the user is limited in the ways they can be expressive and cannot create compositions. Soundstage reproduces the feel of a sound synthesizer studio, with realistic-looking equipment surrounding the user in all directions. However, this project aims to further explore how to take advantage of virtual reality in ways that go beyond replicating real-world objects and interactions. While SoundStage is a great tool to experience expensive real-life equipment in a convenient setting, it is also interesting to examine how VR might be used to create experiences and interactions that cannot exist in real life.

This project aims to explore VR interactions and musical composition techniques to become more knowledgeable in maximally utilizing the opportunities VR provides.

Project Overview

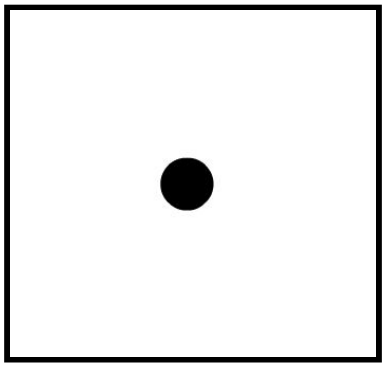
Grid Sequencer

One of the two sandboxes created is called the Grid Sequencer. It consists of several different grids in 3-dimensional space. Each row corresponds to a different note or sound, while the columns represent quantized moments in time. A time bar moves across each grid in synchronization with each other. The user can place cubes in the grid spaces to design an audio loop. When the timebar passes over a cube in the grid, a musical event occurs. Different rotations of the cubes correspond to different effects and parameters. The side of the cube facing in towards the user is the active side. There are three different grids, each with their own corresponding cubes. The cubes match the color of their matching grid's timebar.

Percussion Grid

In the Percussion Grid, each row corresponds to a different percussion sound. Two of the faces of the percussion cubes have dots representing a staccato, and when that face is active, the note has a shorter duration.

Percussion Cubes:

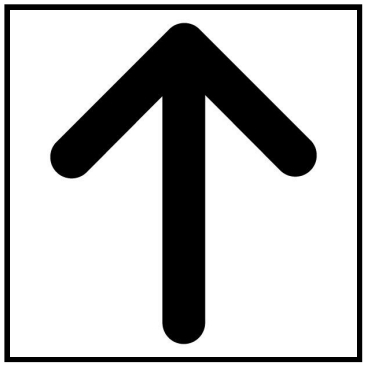


Staccato side of percussion cubes

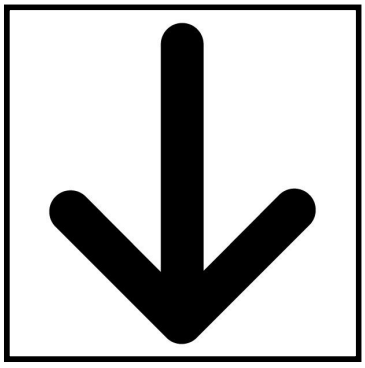
Melody Grid

In the Melody Grid, each row corresponds to a different note in the pentatonic scale. Two faces of the melody cubes have arrows on them. An upwards-facing arrow shifts the pitch one octave up, and a downwards-facing arrow shifts the pitch one octave down. This grid has two layers, and is the only true 3-dimensional grid. It is therefore possible to play two instances of the same note simultaneously, potentially in different octaves.

Melody Cubes:



Upwards arrow: pitch shift up by an octave

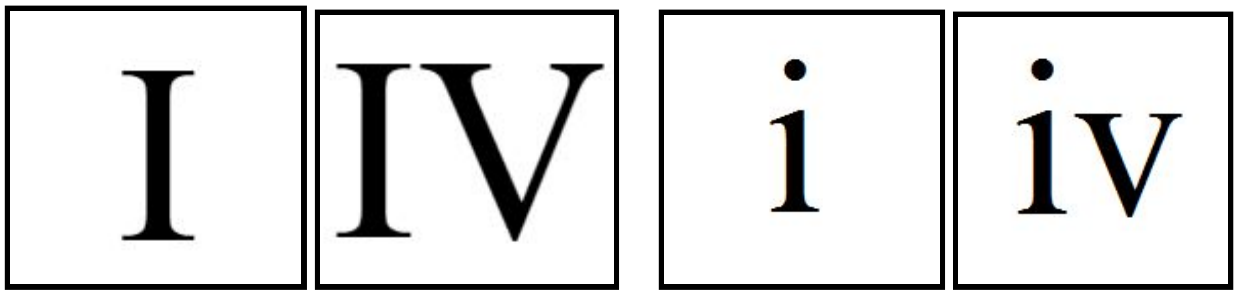


Downwards arrow: pitch shift down by an octave

Chord Grid

The final grid is the Chord Grid, which has two rows, each with a different function. The bottom row determines what chord is played. The cubes that go in this row have a different chord numeral on each face, and by default the chord is played as a whole note. The top row determines the pattern of the chord. The cubes that go in this row have different patterns of dots on their faces indicating their pattern. See below for some examples. A cube in this row only has an effect if there is a cube directly below it to indicate the chord that should play.

Chord Cubes:

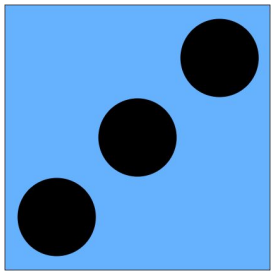


Major One Chord | Major Four Chord | Minor One Chord | Minor Four Chord

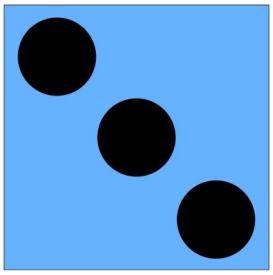
(chord cube faces change depending on major/minor mode)

Chord Style Cubes:

Right Cube Face Rotations:

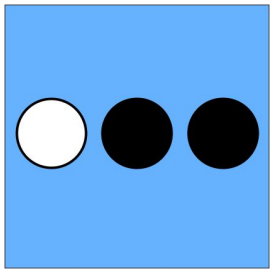


Ascending eighth-eighth-quarter, repeated twice (C major chord would be C-E-G, C-E-G)

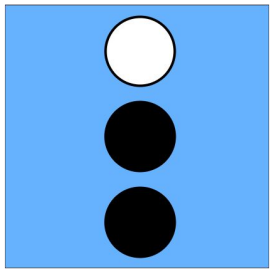


Descending eighth eighth quarter, repeated twice (C major chord would be G-E-C, G-E-C)

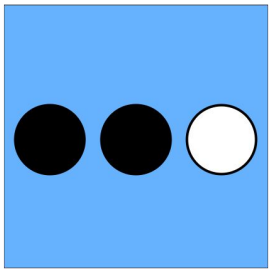
Back Cube Face Rotations:



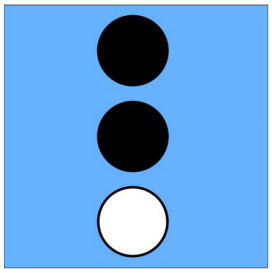
Full chord played as chord
half-quarter-quarter



Top note of chord played as whole note, bottom two notes played as quarter notes



Full chord played as quarter-quarter-half

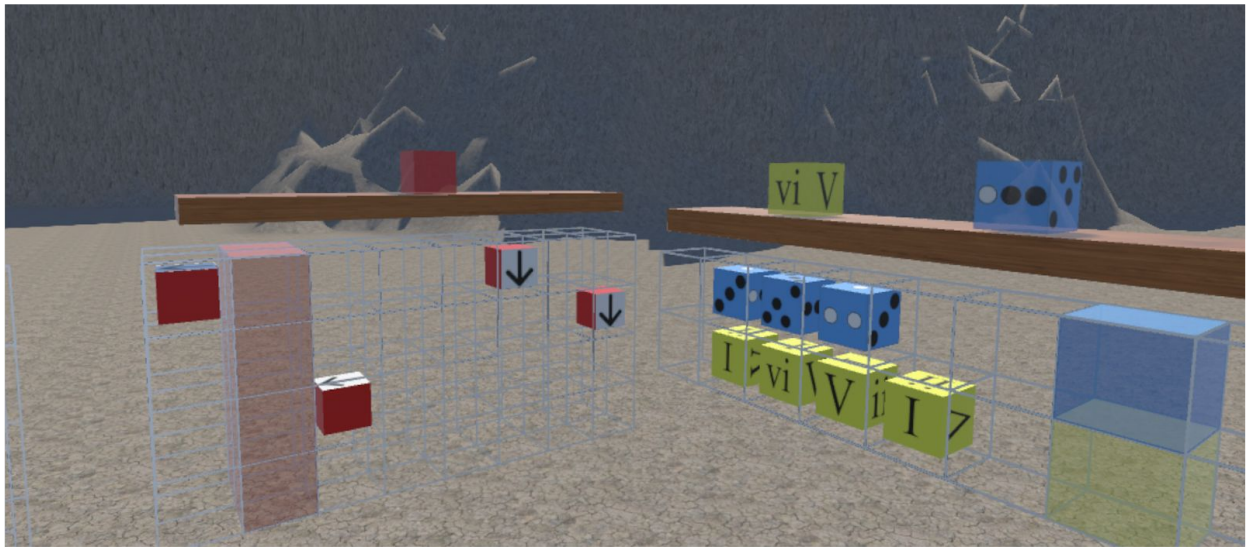


Top two notes of played as quarter notes, bottom note played as whole note

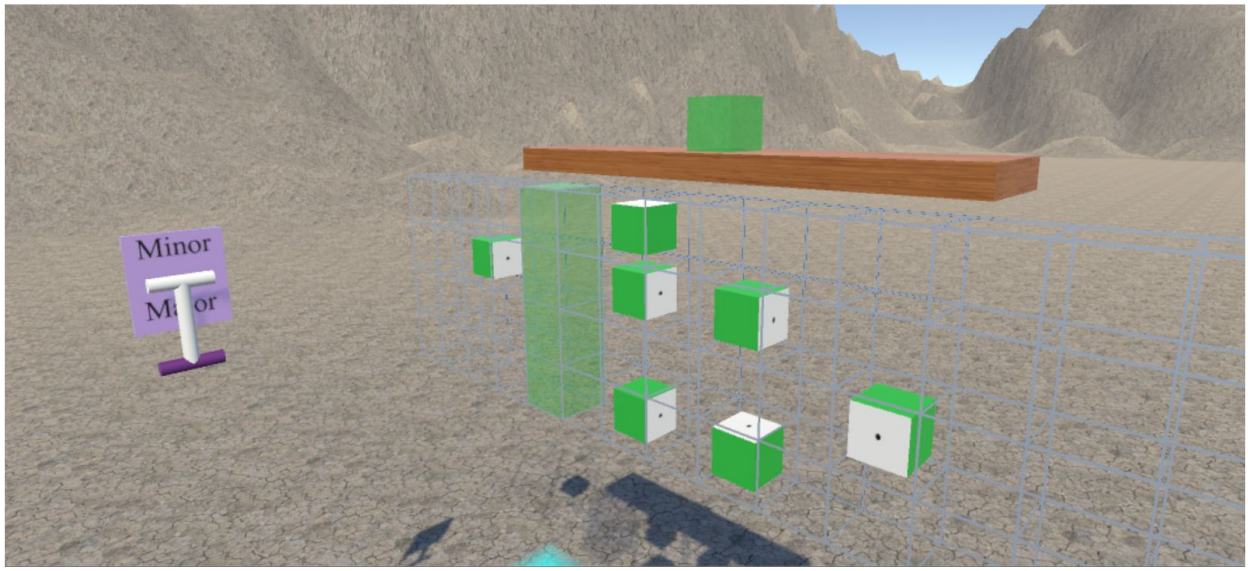
Song Parameters

The Grid Sequencer also has options for changing two global song parameters: tempo and quality (major or minor key). Tempo is controlled by scrolling on the left controller's trackpad, while quality is determined by an in-world lever the user can pull. When in minor mode, the sky turns darker and it rains.

The below image shows the Melody Grid (left) and the Chord Grid (right)



The below image shows the Percussion Grid (right) and the quality lever (left)



Bottle Shaper

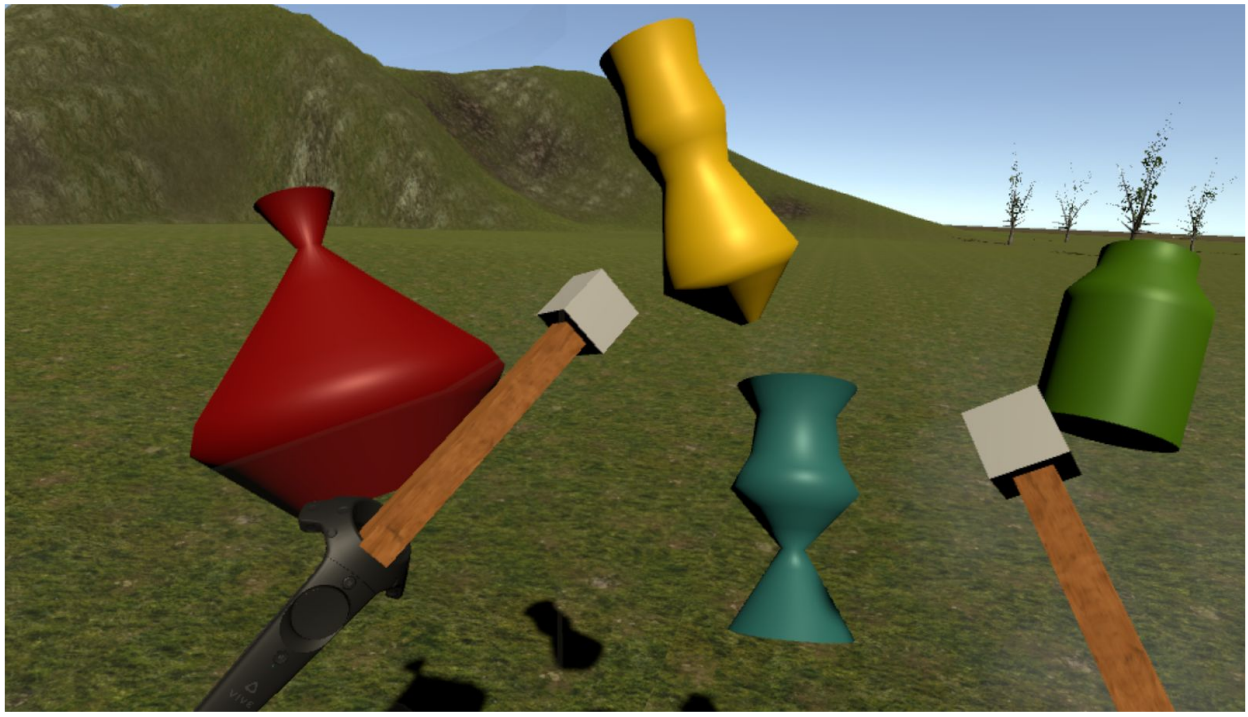
The second sandbox is called the Bottle Shaper. Its main mechanic is allowing the user to reshape bottle objects, where different sizes and shapes correspond to different sounds. The user can press the trackpad button on one of the controllers to create a new bottle. By holding the trigger button, the user can push and pull on the bottle to change its shape and size. The volume of the bottle determines its pitch, which also corresponds to its color. Larger bottles have colors closer to red and produce a lower pitch, while smaller bottles have colors closer to blue and produce a higher pitch. The shape of the bottle is also important. Certain aspects of

the shape are mapped to different audio effects, such as low-pass filters and distortion. The user can move the bottles anywhere in space using the grip button. The bottles do not react to gravity or other forces, so they stay exactly where they are placed. There are two methods to make music with the bottles: hitting them with a drumstick, and placing them in the Fan Zone.

Drumsticks

Drumsticks can be picked up from a table in a corner of the play area by clicking the grip button once. They will stay attached until the grip button is pressed again. When a drumstick hits a bottle, it bounces off and a glass ping wav file of the correct pitch plays. The mallet of the drumstick collides with bottles, but the handle goes through the bottles. The physics was designed this way to prevent the drumstick from getting completely stuck inside a bottle, and to make navigating an area with many bottles easier.

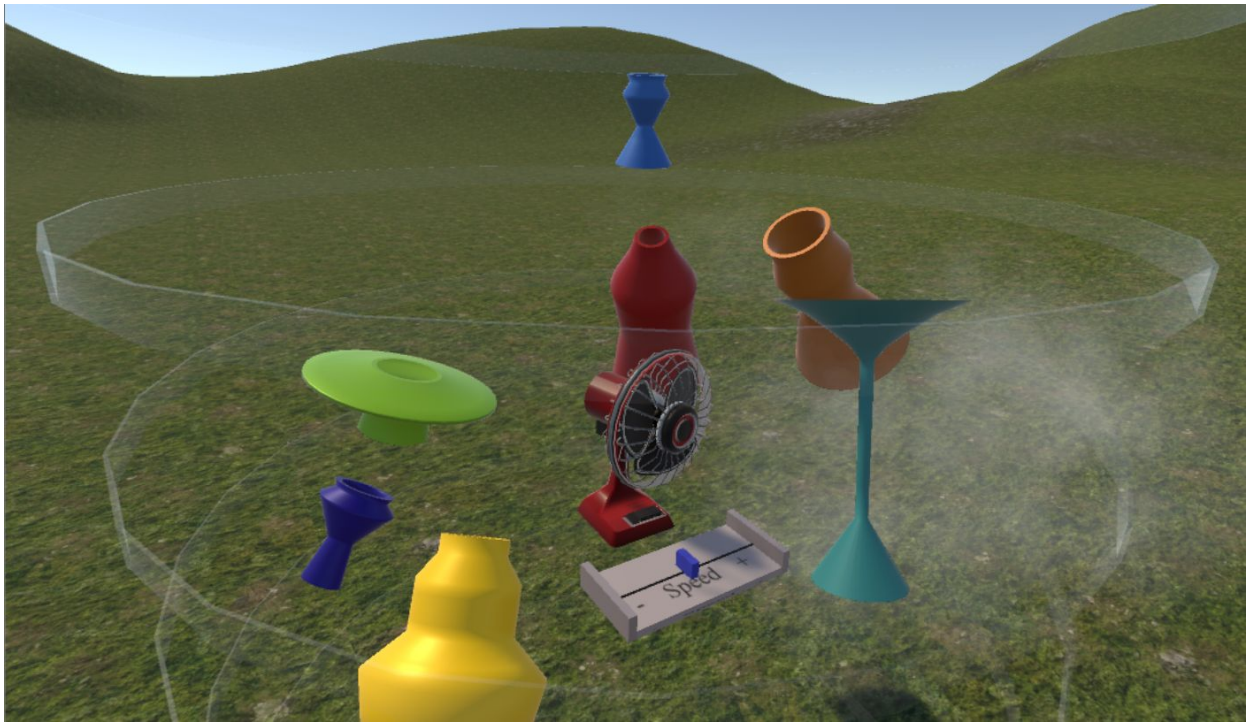
The image below shows several bottles about to be hit with drumsticks.



Fan Zone

One section of the of the play area contains a rotating fan surrounded by transparent rings. These rings enclose an area called the Fan Zone. When the fan blows past a bottle inside the Fan Zone, a blown-bottle sound of the correct pitch plays. The timing of the Fan Zone is quantized, so bottles will only produce a blown-bottle sound on the beat. The user can change the fan rotation speed by grabbing or pushing on the speed slider located below the fan.

The image below shows several bottles inside the fan region, which is enclosed in transparent rings. The fan has a particle system that looks like blowing mist. The speed slider is located below the fan.



Technical Details

General

HTC Vive

The project used the HTC Vive VR system for a full-room experience. The Vive consists of two infrared camera sensors that track the position of a headset and two controllers, all shown in the image below.

Unity

Unity is a 3D game engine that connects to the HTC Vive using the SteamVR Plugin. In Unity, a project consists of different Game Objects with attached components. Components include C# scripts and Audio Sources. Some predefined functions that Unity automatically makes calls to include:

- Awake(), called when the script is loaded
- Start(), called after Awake()
- Update(), called every frame
- FixedUpdate(), called every fixed framerate frame (used for physics updates)
- OnAudioFilterRead(), called when Unity asks for Audio data

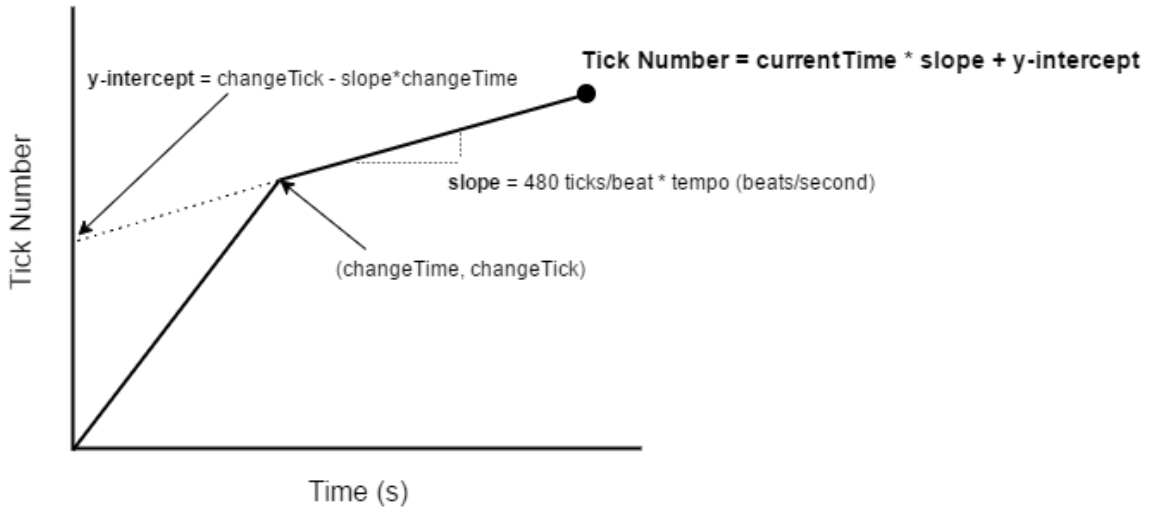
Both sandboxes use a Unity Asset called the Virtual Reality Toolkit (VRTK). This provides scripts that enable grabbing, dropping, and throwing objects. For this project, some minor edits to these scripts were made to customize events and reorder some function calls.

Sequencing System

Both sandboxes used the same sequencing system to play notes at the correct times. This system consists of a Clock, a Scheduler, a Midi Synthesizer, and customized use of Unity's OnAudioFilterRead() function.

Clock

The Clock uses the Singleton design pattern. There exists one instance of the clock that every script can access. The Clock stores the current tempo and the current quality (major/minor). It has an event handler to notify other objects when it changes between major and minor. The Clock also has a list of all the schedulers that exist in order to keep them in sync with each other. The clock keeps time as the program runs, keeping track of the current time in seconds and the current tick. Ticks are a measure of how much musical time has passed. There are always 480 ticks per beat. The Clock has methods that convert from time to ticks and vice-versa. To convert from time to ticks, the tempo must be converted to units of ticks/second, and then a tick offset must be added to ensure tick continuity throughout tempo changes. This is illustrated in the diagram below. In the diagram, the clock is running at a constant tempo (and thus has constant slope) until it reaches changeTime, where the tempo (and therefore slope) changes.



Scheduler

The Scheduler class allows objects to schedule a function to be called at a later instance in musical time (a future tick). Each object that needs to schedule audio has its own Scheduler. The Scheduler has a method PostAtTick() that takes in a function to be called in the future, the tick when the function should be called, and an optional argument to pass into the function when it is called in the future. The Scheduler stores all this information as one Command. The Scheduler has a priority queue of Commands whose functions are waiting to be called, sorted by tick. The method PopScheduledFunctions() returns any Commands whose functions need to be called in some specified amount of time.

Midi Synthesizer

The Midi Synthesizer code provides the actual means for producing audio data. It allows access to the Midi sound bank and Midi Messages such as NoteOn and NoteOff events. Most of this is handled automatically by the CSharpSynth [Sanford]. The function FillWorkingBuffer() must be called to retrieve audio data from the Midi Synthesizer. This code was edited to allow any size buffer instead of a fixed size.

OnAudioFilterRead

Unity has the predefined method OnAudioFilterRead() that is called on a separate thread when Unity asks an Audio Source for data. It can be overwritten to customize the audio that plays, instead of just using Unity's Play() and Pause() methods. An argument containing the current audio data is automatically passed in whenever OnAudioFilterRead() is called. 3D sound spatialization is applied to the data before this method is called, and audio effects are applied

after the method is called. To prevent overwriting the sound spatialization, every Audio Source in the sandboxes constantly plays a wav file of all 1s, which multiplies the custom data.

In OnAudioFilterRead(), any functions that need to be called within that audio buffer are retrieved by calling scheduler.PopScheduledFunctions(). Then, it asks the Midi Synthesizer for as much audio as it needs until a function needs to be called. It then calls the function, and repeats this process until all the functions have been called. This allows functions like NoteOns to be called at the exact correct tick. At the end of each OnAudioFilterRead(), the Clock is told that one of its schedulers has moved forward in time. Because all OnAudioFilterRead() functions are called at once, once all of the clock's schedulers have announced that they've moved forward in time, the entire Clock moves forward in time the same amount. This technique for using the OnAudioFilterRead() function was inspired by materials and information provided by Matt Boch.

Grid Sequencer Scripts

ParameterMappings

This script is attached to each cube, and contains parameter information specific to each cube rotation. Each cube is initialized with a parameter setting (chord, chord style, staccato, or pitch shift). This script stores the values for parameters like note duration, pitch shift, and chord information for each possible rotation of the cube.

CubeFactory

The CubeFactory script enables cubes to regenerate when a cube is picked up from its starting location. It is attached to a cube factory object. Each cube type has a corresponding cube factory object that is the parent of all existing cubes of that type. CubeFactory keeps track of which cube is the source cube (the cube that remains on the shelf). The method sourceCubePickedUp() is subscribed to the InteractableObjectGrabbed event on the source cube. This method instantiates a new cube to become the next source cube. Source cubes do not react to forces and are more transparent than regular cubes.

CubeBehavior

This script handles the physical behavior of cubes. When the timebar crosses a cube, CubePlayAnimation() is started. This is a coroutine that causes the cube to quickly grow and then return to its normal size. This script also has the method CreateCubeCopy() that is called when a cube is held close to a grid. It creates a preview cube object in the grid cell the cube would snap to if released. The parameter setting for the cube is stored in this script. Finally, CubeBehavior has a method ChangeCubeQuality() which is subscribed to the Clock's MajorChangeEvent. This changes the faces of the cube when the song quality is changed. This is important for the chord cubes, because the numerals corresponding to the different chords are different for major and minor keys.

GridBehavior

The GridBehavior script is attached to each grid. This script handles the physical aspects of the grid, as well as its interactions with cubes. In Awake(), grid lines are drawn in 3D space using the Unity asset Vectrosity. Each grid has a list of the CubeFactories it will accept. This prevents percussion cubes from snapping to the Melody Grid, for example. Every FixedUpdate(), GridBehavior loops through its list of acceptable cubes to check if it should lock any to the grid, or if a preview cube should be created. If either of these need to occur, GridBehavior calculates the closest grid cell to the cube and the nearest orientation it should snap to. The script keeps a 3-dimensional array, gridCellOccupation, containing the cube objects that are locked to each grid cell.

GridSequencer

GridSequencer is instantiated by GridBehavior. It handles the audio sequencing for the grid object it is attached to. It initializes a new Midi Synthesizer, and then calls StartSequencer(). This schedules the function _noteOn() or _chordNoteOn(), depending on the grid type, to be called at the beginning of the next musical bar. This keeps all grids synchronized. _noteOn() and _chordNoteOn() schedule themselves to play again a number of beats in the future specific to the grid. They are therefore called every time the timebar moves to the next column in the grid. _noteOn() checks whether there are currently any cubes in the current column, and if so plays notes specific to the rows and parameter mappings of the cubes. _chordNoteOn() follows a similar idea. However, some chord styles require notes to play at different times within the same timebar. To accomplish this, _chordNoteOn() schedules the function _basicNoteOn() for each note indicated by the chord style cube.

LeverScript

This script controls the lever's rotation to ensure that it stays in bounds. When the lever is released by the controller, this script sets the Clock's major/minor boolean to the correct value.

Bottle Shaper Scripts

CreateBottleOnClick

This script instantiates a new bottle object when the controller's TouchpadPressed event is called.

BottleHolder

This script is attached to the parent object of the bottles. It ensures that only one bottle interacts with a controller at a time.

BottleBehavior

This script is attached to every bottle object. It procedurally creates the initial bottle mesh by positioning all the vertices, and then connecting these vertices with triangles. The vertices are arranged in rings going up the bottle. This script also handles bottle reshaping. It listens for the controller to send an event that the trigger button has been pressed. It then checks if the controller is within range of the bottle. If it is, it sets isInteracting to true and attaches to the ring of the closest vertex. In Update(), if the bottle is interacting, it changes the positions of its vertices according to the position of the controller. The bottle always remains symmetric with respect to its central axis. Rings' radii are affected on a linearly decreasing scale the farther they are from the ring attached to the controller. This script sets the color of the bottles according to volume, and sets audio effects according to shape. For example, the depth of the chorus filter is based on how many times the bottle alternates between increasing and decreasing in radius.

BottleMusic

This script handles playing the glass ping wav file in the function OnCollisionEnter(). It sets the speed of the wav file to produce the correct pitch.

BottleSequencer

Each bottle has a BottleSequencer. This lets the blown-bottle sounds originate from the bottle location in 3D space. The Fan Zone is divided into sectors. In Update(), BottleSequencer checks if the bottle is inside the Fan Zone. If so, it then checks if the bottle is in a new sector that has not yet been scheduled. If this is the case, it schedules TryNoteOn() at the next tick corresponding to that sector. TryNoteOn() checks if the bottle is still present in that sector, and if so plays a Midi note of the correct pitch.

FanBehavior

This script rotates the fan at the correct speed during calls to FixedUpdate(). Start() initializes the rotation to keep it aligned with the BottleSequencer instances.

TempoSlider

This script sets the Clock's tempo to match the position of the speed slider every FixedUpdate().

User Testing Setup

To test the two music composition sandboxes, 16 participants playtested both and completed a survey afterwards. Half the participants playtested the Grid Sequencer first, and the other half playtested the Bottle Shaper first. The purpose of this was to control for any bias that could result from the first sandbox being more novel than the second, or from the

participants being more tired later in the session. Before playing each sandbox, each participant read a page of instructions detailing the controls and aspects of the sandbox. They played each sandbox for 10 minutes, or less time if they felt finished before 10 minutes had passed. Finally, everyone completed a survey about their experience. Throughout the playtest session, we took notes on comments the participants made while playing, and any interesting behaviors and trends we observed.

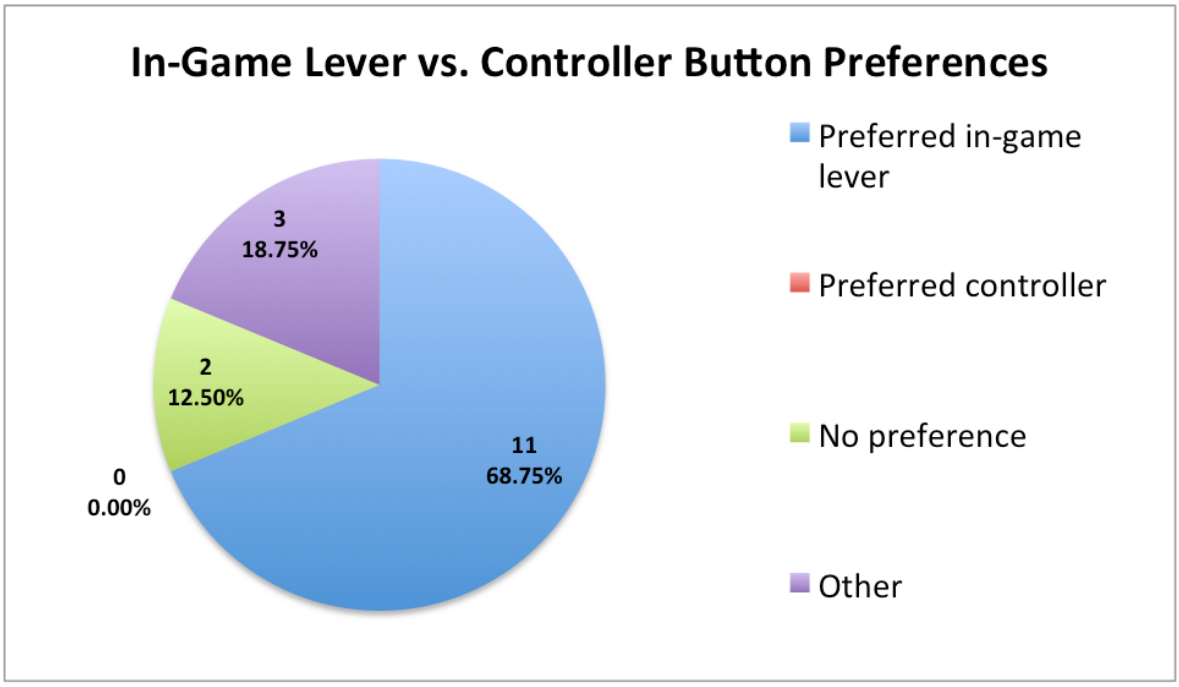
Results

See the Appendix for the full survey and results summary, including comments.

Discussion of Results

Importance of Physicality

The survey asked the participants if they had a preference between the using the in-world lever to make global song changes (major/minor) versus using the controller for global song changes (tempo control). The graph below shows the results.

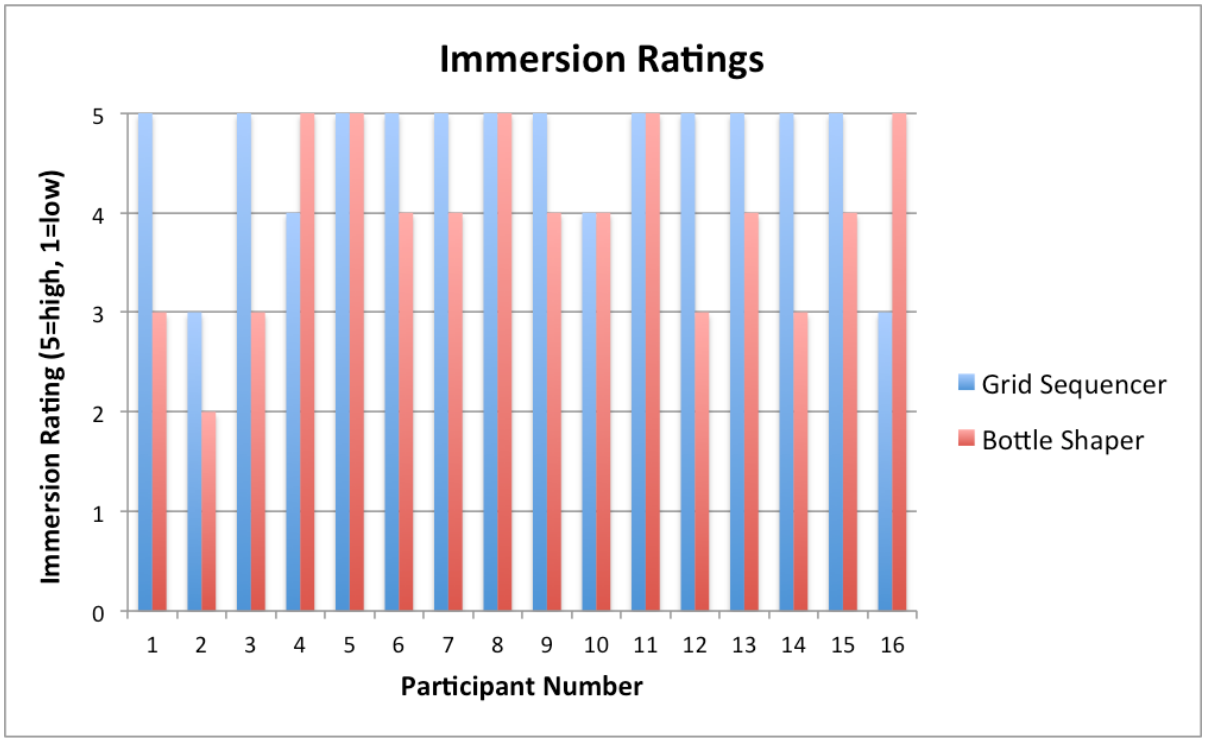


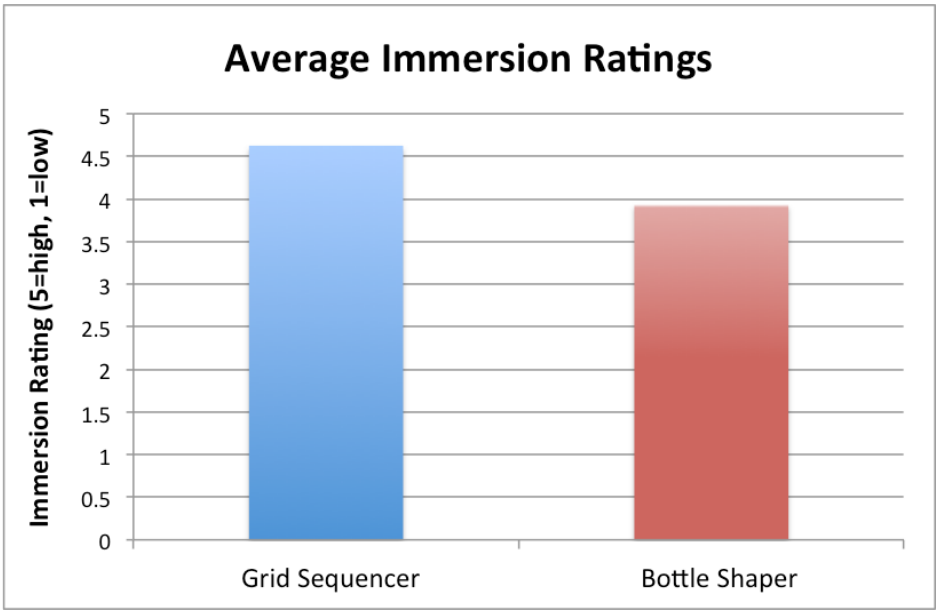
68.8% of participants preferred using the physical lever, while no participants preferred using a button on the controller. This shows that in virtual reality, making interactions feel physical is important. Even though the controller button provided more convenience than a lever that require the user to be in a certain location, the enjoyment of using an in-world object outweighed the inconvenience. One participant commented that they did not know they could

change the speed with the controller, despite indicator text displayed on the controller during runtime. This suggests that in-world objects are more noticeable and tempting to interact with than controller buttons. This is further supported by one user's question while playing the Grid Sequencer. The user wanted to know if there was a way to delete cubes, but phrased their question by asking if there was a garbage can. It seems that because virtual reality simulates the real world, using in-world objects feels like the most natural way to accomplish tasks.

Immersion Ratings

The survey asked the participants to rate how immersed they felt in the Grid Sequencer world and the Bottle Shaper world on a scale from 1 to 5. 1 is not at all immersed, and 5 is completely immersed. Below is a graph showing each participant's ratings, and a graph comparing the average ratings of the two sandboxes.





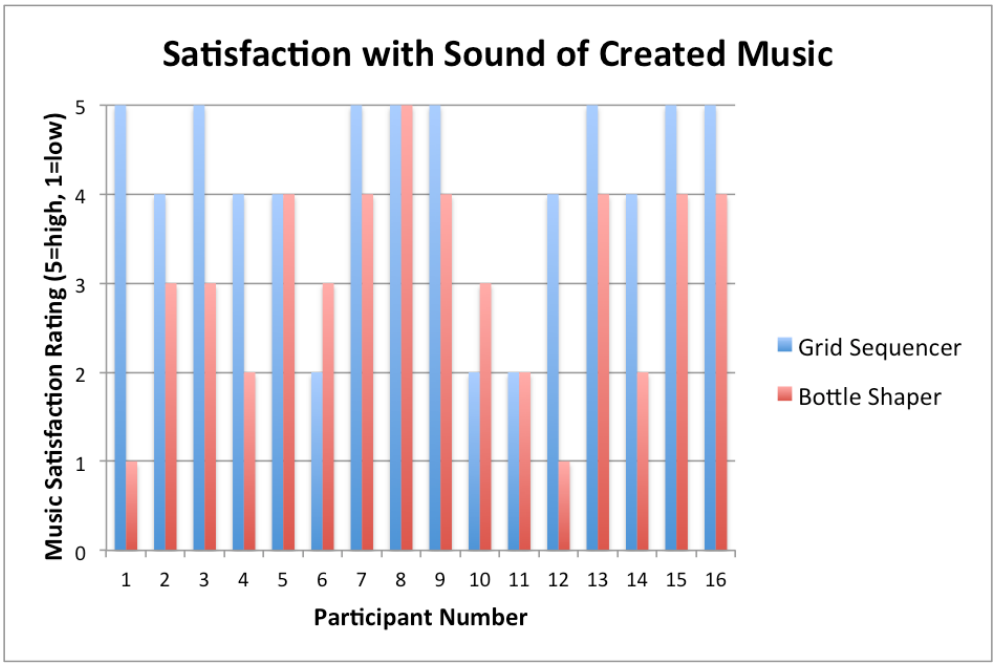
A 1-tailed, paired t-test shows that the participants felt significantly more immersed in the Grid Sequencer world than the Bottle Shaper world. The p-value is 0.0145, which is less than the threshold of 0.05. The survey asked for optional comments after this question, which provides some insight into this result. Six commenters cited inconsistencies or unexpected behavior with the bottle and drumstick physics as factors that lowered their feeling of immersion in the Bottle Shaper world. Some commenters found it strange that the drumsticks were affected by gravity but the bottles weren't. Several comments talked about the drumsticks going partially through bottles and getting stuck, and one participant found it odd that bottles could go through each other. One participant was disappointed they couldn't break the bottles, and that this seemed unrealistic. While playing, two participants attempted to place their first bottle on the drumstick table, as would be natural to do in real life, before realizing that the bottles could float.

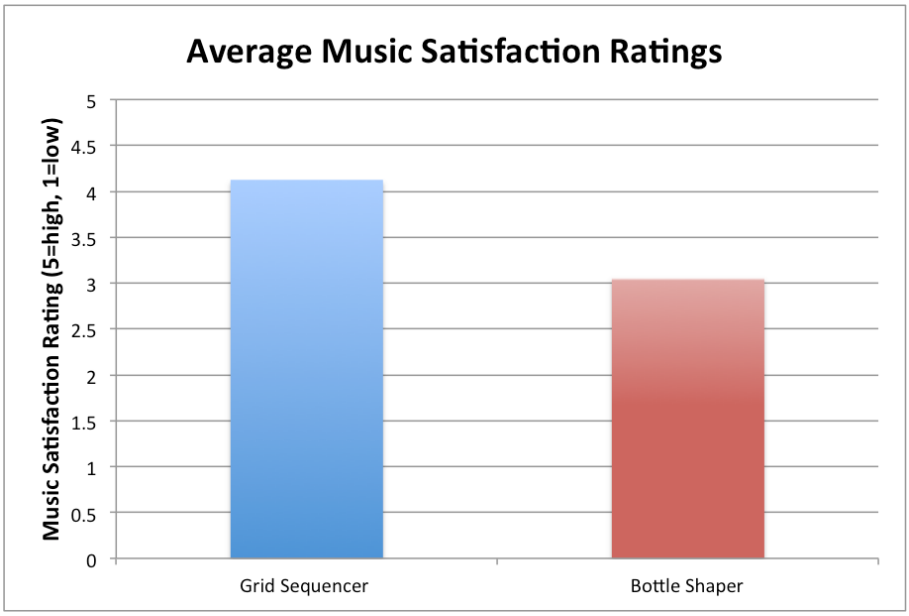
All of this suggests a few interesting ideas. First, because virtual reality feels real, anything that seems would be possible in the real world is expected to be possible in the virtual world. If something that seems possible (like breaking bottles) is not permitted, that breaks immersion for the user. This is further supported by the way users interacted with the Fan Zone. When placing a bottle in the path of the fan, several participants tried to rotate the bottle to see if it produced a different sound. One participant mentioned they expected volume to change depending on the bottle's distance from the fan. These are all features that would naturally exist in real life, so the participants expected them in the virtual world. When their expectations were not met, they felt less immersed. The next main idea is that consistency is important. Many participants seemed fine with the idea of bottles being able to float, but when mixed with drumsticks that cannot float, something seems wrong. Finally, any glitches in physics are a big deal in virtual reality. Many participants were confused when their drumstick became stuck in a bottle, or when their drumstick started moving in unexpected ways. In fact, when asked for their response to any unexpected, nonphysical events, 11 participants checked off the choice "Sometimes it made me feel less immersed in the world."

There were two participants who reported feeling more immersed in the Bottle Shaper world than the Grid Sequencer. One of these participants said this was because the task of actively reshaping bottles pulled them in more than just moving around cubes. Overall, the most popular choice for the participants' favorite activity in the Bottle Shaper was making different bottle shapes, at 37.5%. This suggests that more complex tasks that allow for more creativity add to an immersive experience, but not at the cost of losing consistent physics.

Satisfaction with Sound of Music

The survey asked participants to rate their level of satisfaction with the music they created in each sandbox. Their responses can be seen in the graph below, as well as the mean response for each sandbox.



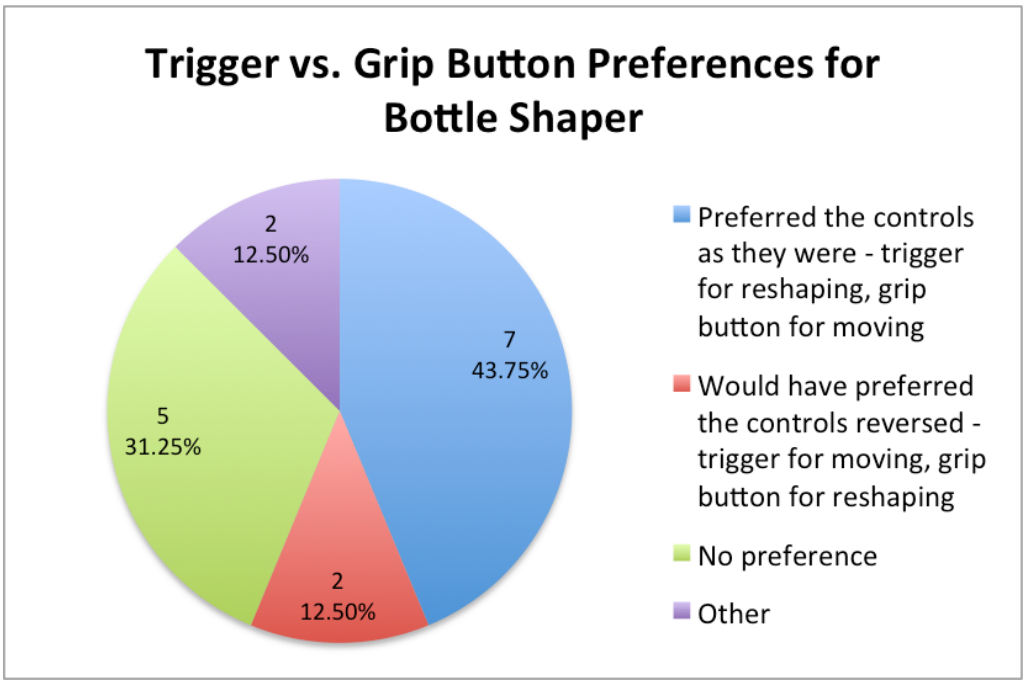


A 1-tailed, paired t-test shows that the participants were significantly more satisfied with the sound of the music they created in the Grid Sequencer than the Bottle Shaper, with a p-value of 0.0032. The data did not show a correlation between music experience level and the difference in music satisfaction rating between the sandboxes, with r = 0.205. However, there weren't enough participants in each category of musical background for a conclusive correlation test. A common reason participants provided for being less satisfied with the Bottle Shaper is that it was difficult to produce bad-sounding music in the Grid Sequencer, whereas it was easy to produce simultaneous dissonant pitches in the Bottle Shaper. One participant commented that they were more interested in individual sounds in the Bottle Shaper. However, in that sandbox participants seemed to have trouble creating a cohesive song. One user mentioned that they did not have enough time to make many different notes in the Bottle Shaper. Another said that they tried to separate bottles from each other, but the fan still played them at the same time. This is due to the way fan sequencing is quantized. It is possible that making the fan sequencing more intuitive and giving the users more time to interact and explore could result in increased musical satisfaction.

These results show that there is a balance between giving the user creative freedom and automating sounds to ensure good-sounding music. The Grid Sequencer automated the music enough so that dissonance was difficult to achieve. However, when asked, 8 users stated that they would have preferred more control in the Grid Sequencer. In the Bottle Shaper, 9 users would have preferred more control and 3 would have preferred less control. From the comments about the Bottle Shaper, it seems that some users would have liked less control in the pitches that were possible, but more control in the timing of the Fan Zone.

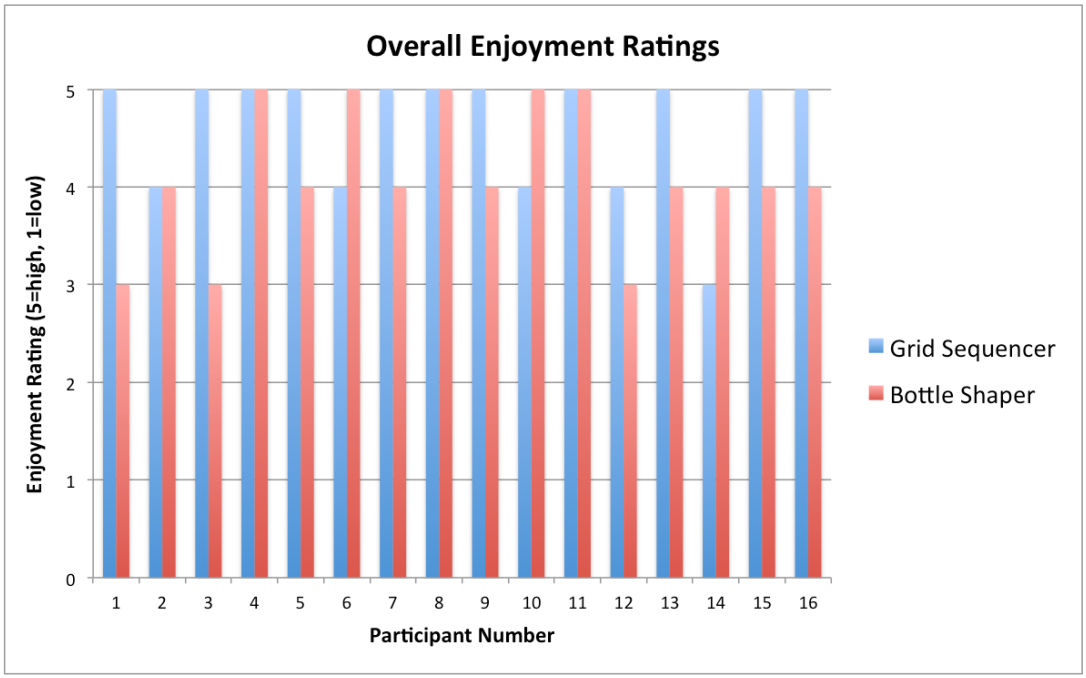
Button Choice

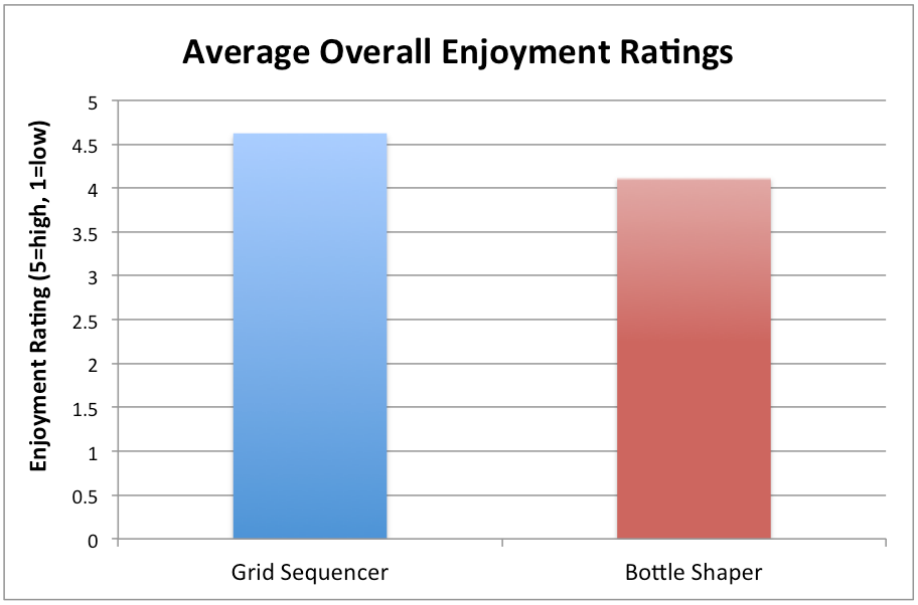
The survey asked participants to select their preference regarding button choice in the Bottle Shaper. The purpose of this was to investigate how intuitive people find using the grip button to grab objects. A graph of the results is below.



While 75% of users preferred the controls as they were or had no preference, those who would have preferred the controls reversed had strong preferences. Two participants stated that the grip button felt unintuitive to them, and this was their main reason for feeling less immersion or less satisfaction with the music creation process in the Bottle Shaper. This shows that the choice of controls can have a large impact on some users' feeling of immersion and enjoyment. In virtual reality games and sandboxes, it may be important to either playtest several different button setups to find the most optimal one, or give users the option to customize their own button setup.

Overall Enjoyment Ratings





Overall, participants preferred the Grid Sequencer slightly more than the Bottle Shaper, with a 0.5 difference between their average enjoyment ratings. This makes sense, given that in the Grid Sequencer world, participants felt more immersed and were more satisfied with the sound of the music they created.

Suggestions for the Future

The feedback from the playtest sessions inspired some ideas on how to improve the sandboxes, and some thoughts about other interactions and experiences that might be interesting to explore in virtual reality.

Grid Sequencer

A few modifications and additions could be made to the Grid Sequencer. Tempo control could be changed to a physical slider, and volume control dials could be added to each grid, as one comment suggested. Since participants generally desired more control in the Grid Sequencer, the user could be allowed to customize the instrument sounds for each grid. Another possibility is to let the user compose multiple, separate loops on the same grid. The user could manually switch the loop that is playing, or choose to automatically switch between loops. This would prevent the music from getting too repetitive. This feature could be executed by using the third dimension of the grid, with only the front layer containing actively-playing cubes.

Bottle Shaper

The survey results inspired several changes and additional features for the Bottle Shaper. First, the drumstick physics should be fixed to produce consistent behavior. If drumsticks colliding with the bottles continues to cause issues, it is worth exploring the approach SoundStage takes with drums. The drumsticks would physically pass through the bottles, but the controller would buzz on the collision to give tactile feedback. Bottle creation could become more physical by making a Bottle Creation Machine.

The next series of large changes involve the Fan Zone. From comments and observations, this sandbox would be more immersive if it were more reactive to the different bottle positions and orientations in the Fan Zone. First, sequencing could be made unquantized, or a switch could let the user switch between quantized and unquantized mode. This would make the timing of the bottles more realistic. Volume of the blown-bottle sound could decrease as distance from the fan increases, and the bottle's rotation could be taken into account when calculating a note's envelope.

One small tweak is to widen the range of allowed pitches the bottles produce, since some commenters were disappointed in the pitch limitations. Since users weren't as satisfied with the music they produced in the Bottle Shaper, the sandbox could be modified to only allow pitches that sound good together, instead of being able to create every possible pitch.

Potential Future Explorations

There are many possibilities for different types of interactions in virtual reality that could be used for music composition games. One idea is to better utilize the area outside of the playzone. While the user can't physically access this area while playing, it can still affect their

experience. For example, the user could slide an object down a long chute that extends past the play area towards some large object in the distance. A musical and visual event could occur when it reaches its destination. Something like this could give the user a better feeling that they are in a large space. For similar reasons, teleportation would be interesting to explore. This would permit direct interaction with areas normally outside of the playzone, giving the user access to more space.

Conclusion

Virtual reality is a new technology that has yet to be fully explored. Several very different VR experiences are available that focus on aspects of artistic creation, music experience, and music composition. This project aimed to further investigate the features and elements that work best for a virtual reality music composition sandbox. It continued previous work from the summer to complete demo versions of two different sandboxes. It then focused on playtesting both sandboxes and asked the participants questions to attempt to gauge what worked well, what did not work well, and what was most important to the users. The results show several patterns. First, the use of in-world objects whenever possible is important. Users enjoy interacting with their world more than their controller. Next, some of the most important factors for immersion are consistency in physics and object interactions, and how closely the user is able to affect their surroundings to meet their expectations of what would happen in real life. Finally, in a music composition sandbox, it is important to find the right balance between user control and automation in the system. The user should feel like they created something unique, but the composition process should also not be too difficult or complicated. From the feedback received, several specific changes and additions could be made to each sandbox to improve the user experience. Overall, given the small fraction of time virtual reality has been available compared to other video game mediums, there is much more to create and discover. It is important that new interactions and ideas continue being explored and evaluated in order to take full advantage of the benefits of virtual reality.

Appendix

Link to survey results:

https://drive.google.com/file/d/0B-IIR9f_pPDLbWVuRTZuVzQyczQ/view?usp=sharing

Works Cited

Boch, Matt. “unity midi / synth / scheduler?” Message to Eran Egozy. 20 June 2016. Email.

Fitterer, Dylan. *Audioshield.* Dylan Fitterer, 5 April 2016. Steam.

Google. *Tilt Brush.* Google, 5 April 2016. Steam.

Hard Light Labs. *SoundStage*. Hard Light Labs, 7 July 2016. Steam.

Harmonix Music Systems. *Harmonix Music VR.* Harmonix Music Systems, Oct 2016. Playstation 4.

Sanford, Leslie. (2014) *C# Synth Project*. [Computer program]. Available at http://csharpsynthproject.codeplex.com/ (Accessed 20 June 2016).