Senior Project
VR Orchestra App: Violin Prototype for Desktop

Jocelyn Dunkley

Advisor:
Aline Normoyle
Swarthmore College, Department of Computer Science

Submitted in Partial Fulfillment of the Requirements of the BA in Computer Science
Bryn Mawr College
Spring 2020
Abstract

Primary and secondary schools’ arts budgets have continually been cut because of their status as non-essential subjects [1]. However, music education has been proven to be extremely beneficial for child development as it fosters discipline, increases creativity and memory capacity and contributes to higher performance in academics [2]. This project seeks to give children a comprehensive personalized music education experience while cutting costs for school districts who may not have the funds to buy instruments and refurbish them. The VR Orchestra App provides music teachers a supplemental tool to teach about instruments in orchestras. After using the app, children will be able to identify instruments visually and aurally and remember core characteristics about each instrument and about the orchestra as a whole. This project built a fully-functioning prototype of the violin section for desktop use. In the future, the app will be developed for the Oculus Rift and will be fully implemented for all thirteen instruments in the orchestra and include quizzes and mini-games.
Acknowledgements

I would like to thank Dr. Jessica Linker for sparking my interest in using Virtual Reality for educational purposes. I also want to thank my Advisor, Professor Aline Normoyle, for providing me with the guidance, expertise and validation to help me build this project and execute my vision. And finally, I would like to thank all my friends and family for motivating and encouraging me to work on this project.
Contents
1. Introduction..................................................................................................................................7
2. Related Work...............................................................................................................................8
3. Motivation and Background......................................................................................................10
4. Methodology..............................................................................................................................11
  4.1 Architecture Design.....................................................................................................11
  4.2 Parts of the violin subsection.......................................................................................13
    4.2.1 Overview.......................................................................................................13
    4.2.2 Design and Implementation Choices............................................................14
  4.3 Hear the violin subsection...........................................................................................15
    4.3.1 Overview.......................................................................................................15
    4.3.2 Design and Implementation Choices............................................................16
    4.3.3 Animating the bow........................................................................................16
    4.3.4 Lining up the audio file with the bow animation..........................................17
    4.3.5 Changing strings...........................................................................................18
  4.4 Play the violin subsection............................................................................................20
    4.4.1 Organization of play the violin.....................................................................20
    4.4.2 The Practice Note Game...............................................................................21
    4.4.3 Accuracy-based Note Game...........................................................................23
5. Results and Analysis..................................................................................................................25
6. Future Work...............................................................................................................................26
7. Summary ...................................................................................................................................27
References .....................................................................................................................................28
1. Introduction

Music and the arts have been deprioritized in public schools as educational budgets have decreased [1]. As a result, many music programs have been cut or severely underfunded. However, music education has been proven to be extremely beneficial for child development as it fosters discipline, increases creativity and memory capacity and contributes to higher performance in academics [2]. The goal of this project is to develop a VR music education application that can maximize the impact and learning opportunities for students in schools that do not have many resources for music and the arts. This includes most schools in the US. Schools with thriving music programs are usually in wealthier neighborhoods that depend upon community donations to fund the program. Buying and maintaining musical instruments can cost tens of thousands of dollars. There are usually not enough instruments for children to properly learn or experience playing each one. As an alternative, VR equipment has an upfront cost but can be used for years in any academic subject in a school in addition to music classes. Class sizes are also increasing due to lack of funding and teachers so individualized attention from one teacher for dozens of children is practically impossible. Learning an instrument requires many hours of supervised learning which is not possible with current class sizes. Therefore, this project provides an alternative application that can give children a comprehensive background about instruments in the orchestra. I hope the users of the app will be inspired in the future to learn how to play an instrument or to continue learning about music.

Learning an instrument is very challenging and requires tens of thousands of hours of practice as well as supervised teaching to master. Virtual Reality offers a solution to receive individualized attention in one’s music education in order to progress faster and learn more efficiently. VR as a teaching tool has been shown to be extremely beneficial for the process of learning and retaining information. Krokos, Plaisant and Varshney (2019) found that “virtual memory palaces in a HMD [head mounted display] condition provide a superior memory recall ability compared to the desktop condition,” with an 8.8% better memory recall rate [3]. Learning through VR also increases the effectiveness of contextual learning and helps users make connections to past experiences or topics learned. The improved understanding and absorption of material using VR tools has led to better grades and test scores for children in science classes [4]. However, there has not been much research conducted on the effectiveness of VR in music classes.

Virtual Reality is an interactive medium and allows the student to be more engaged with the material. Ray and Deb (2016) demonstrated that children prefer VR learning experiences over traditional teaching methods. VR tools increase children’s information retention, engagement with the material and participation in class [5]. The direct contact to the material makes the learning experience more engaging and makes the student more excited to learn and participate. Finally, Virtual Reality brings modern technology into the classroom and students
are exposed to new ideas and ways of conveying information. When students are comfortable with modern technology, they are equipped with skills to learn and work with technology in the future.

In conclusion, there are obvious benefits and opportunities for bringing VR to music education such as increased participation and deeper understanding of content. With the VR Orchestra App, children learn about instruments in the orchestra and understand the differences between each type and how to play each instrument. This will provide useful background information in their music classes and develop their musical education.

2. Related Work

Music-related Virtual Reality (VR) and Augmented Reality (AR) applications have become wildly popular over the last three years in both entertainment and education [6]. Major music labels such as Universal Music Group and Sony Music Entertainment have developed applications in partnership with VR companies like Redpill VR to bring live events to music fans. They have developed VR film projects and sold tickets to VR concerts where major artists perform [7]. VR and 360 degree videos have also been incorporated into music videos. One of the most popular VR music videos is the Gorillaz song “Saturnz Barz.” Maroon 5’s “Summer” cover by Roomie features 360 degree video of various sites in London and the Weeknd’s “The Hills” remix with Eminem is a combination of VR and 360 degree video and allows the viewer to experience a first-person view of the song’s storyline [8].

There have also been advances and a growing interest in VR/AR music education applications to improve skills in note and rhythm training [9][10][11]. Most applications focus on helping individuals teach themselves how to play instruments or other musical skills. However, few applications for teaching music have been developed for the classroom. This project addresses this gap. Using the VR Orchestra App, children can learn about the parts of different instruments, listen to the instrument play in different genres, and simulate playing the instrument. The current prototype demonstrates these features using the violin.

Many of the VR/AR applications that have been developed for learning how to play instruments individually have been inspired by the design and gameplay of Guitar Hero. Guitar Hero is a music rhythm game where users use guitar-shaped controllers to simulate playing the guitar. They press buttons on the “frets” of the guitar in time with the music on screen to score points.

Music Everywhere is one such app that uses similar gameplay as Guitar Hero. It is an Augmented Reality piano app that guides users through playing piano in collaborative settings.
and virtual band environments as seen in Figure 1 [12]. The app is available to use on an Augmented Reality-compatible phone or with the Microsoft HoloLens. It is in a Guitar Hero format where notes drop down from the top of the screen over a specific key of the piano and guides the user to play a song. It also teaches music theory through animated demonstrations with a hand playing certain keys.

![Figure 1: Music Everywhere guides users through a song.](image)

Users can also learn to improvise in a variety of musical genres and practice with a virtual band. The app teaches how to develop strategies to know which chords are appropriate to change to and anticipate changes while improvising. Music Everywhere uses some of the same techniques to teach music as in this work. This project uses the same guided style in the *play the violin* section’s note games.
Figure 2: Augmented Reality-based feedback on violin finger placement using a depth camera.

Research has also been conducted to help those teaching themselves violin receive regular feedback as they practice. De Sorbier, Shiino and Saito (2012) developed an Augmented Reality system that helps a player know “where to correctly press the strings on the fingerboard and how to perform the bowing movement by displaying virtual information on a screen,” using a depth camera and an Augmented Reality screen, as seen in Figure 2 [13]. The system captures the player’s positioning and violin through a depth camera and reconstructs the violin through a point cloud and color capturing. It then compares the positioning of the violin versus the player and offers adjustments. This research and application is similar to the future VR version of play the violin section in this project because the app will use the position and angle of the Oculus Rift controllers to measure if the player is in the correct position or “playing” the correct string. Although the application will not be as accurate as this research since the user will not be playing on a real violin, these techniques will be helpful to know how to track, encourage, and correct the player to improve their positioning.

3. Motivation and Background

For this project I implemented a prototype of one of the instruments, the violin, as a desktop application to illustrate the look and function of the application. When the user first opens the app they are greeted with a semi-circle of instruments they can choose from. They can explore what each instrument looks like from any angle. When they are done exploring they can click on a particular instrument to learn more about it. When the user clicks on the violin a menu pops up where the user can learn about different aspects of the violin.

There are three options the user can choose from: parts of the violin, hear the violin and play the violin. The parts of the violin section is a walkthrough tutorial of each part of the violin with narration about the uses, material and functionality of the part in question. Next, the hear the violin section teaches the user about the versatility of the violin and its use in different genres
such as classical, country/fiddle and jazz. When a genre is selected, a song will begin playing and the violin will “animate” in sync with the song.

Finally, the play the violin section allows the user to simulate playing the violin. The play the violin section was developed for desktop and keyboard use for this project but in further versions it will be updated to be compatible with the Oculus Rift. After a tutorial of how to use the controllers and an overview of how to read music if the user doesn’t know, the user can “play” Twinkle Twinkle Little Star, once first to get used to the controls of the game and then a second time as an accuracy-based game.

4. Methodology

In the following sections, the overall architectural design of the project will be discussed as well as in depth explanations of each violin subsection’s purposes and development and algorithmic designs and implementations. The VR Orchestra App is built using the Unity Game Engine. The architecture of the system consists of subsections corresponding to each of the instruments in the orchestra. For this project the violin prototype and its subsections will be discussed. However the design of the application easily supports scalability to add more instruments. Each instrument follows the same subsection design with different content. The instruments do not interact with each other so they can be built separately using the given roadmap in Figure 3 and then integrated into the app.

4.1 Architecture Design

Figure 3: Block diagram of VR Orchestra app violin prototype.
The architecture design, shown in Figure 3, consists of three primary components: *parts of the violin*, *hear the violin* and *play the violin*. The app is developed with Unity 3D, a game development software, and is intended to be used as a desktop application in this project. First, the user is met with a welcome menu that leads them to the *learn about instruments* scene. In the scene the user sees a first-person view of a semi-circle of instruments surrounding them with a prompt to choose an instrument to learn more about it. The user can look and move around to explore the world. This project only implemented the full functionalities of the violin so the grey blocks in Figure 4 represent placeholder instruments that have yet to be developed.

![Figure 4: The violin in the learn about instruments scene next to two placeholder instruments](image)

When the user is ready to move on they will move to the violin in the scene and click on it, revealing a menu which will lead to the subsections of the violin as shown in Figure 5.
4.2 Parts of the violin subsection

4.2.1 Overview

The parts of the violin subsection is a tutorial overview that teaches users about the functionalities of each part of the violin in order to understand how each part fits into each other. Every piece of the violin is essential but it is difficult for a new student to understand their uses. So this section provides a visual representation for the user to understand the purpose and use of each part and how they work together.

The tutorial consists of an audio narration that I wrote and recorded as well as a narration transcript next to the part in question, seen in Figure 6, in order to account for users with aural and visual learning preferences.
The *parts of the violin* subsection has two tutorial sections. There is a mandatory initial tutorial that consists of ten sections. It provides users a basic overview of each part of the violin, such as the chin rest, tailpiece, strings, bridge, and other parts. Then the Extended Tutorial menu is displayed from which the user can choose a part again to learn more detailed information about it, seen in Figure 7, or go back to the *learn about instruments* scene.
4.2.2 Design and Implementation Choices

**Annotation class:**
- Name of part //string
- Narration //audio file
- Animation from one part to another //animation
- Narration transcript //string

**PlayInitialTutorial class:**
- Displays Annotations in order
- Annotations stored as an array

**PlayExtendedTutorial class:**
- Displays Annotation when user clicks on a part
- Annotations stored as a dictionary where key = part name and value = Annotation object

Figure 7: Extended Tutorial menu. Users can click on a part name and learn more detailed information about it.

Figure 8: Block Diagram of the *parts of the violin* subsection’s architectural design.
Figure 8 shows the architectural design of the parts of the violin section. The Annotation class is the base object used in this section. It is used when going through the initial and extended tutorials. The Annotation class contains the fields necessary for each part of the violin, such as the part’s name, the narration audio file, the animation that pans the camera from the previous part to the current part, and the narration transcript to display to the user. The Annotation class is used differently for the initial and extended tutorials. In the PlayInitialTutorial class, an array of Annotation objects is given that was pre-populated in the order to display each part. The class iterates through the array and displays the information about each part of the violin like in Figure 6.

On the other hand in the PlayExtendedTutorial class the user can click on any part, shown in Figure 7, and the corresponding Annotation information is displayed. Instead of an array, I used a dictionary that had the key as the part name and the value as the Annotation object. I created a method for each part of the violin and mapped it to its corresponding button in Figure 7. When a part’s button is clicked the PlayExtendedTutorial method is called for that part and the specific Annotation information is displayed. Additionally, the narration audio file and an animation from the starting screen in Figure 7 to the part is started.

4.3 Hear the violin subsection

4.3.1 Overview

The purpose of the hear the violin subsection is to show the user the violin’s versatility and use in various genres. I used three genres and songs to demonstrate this; classical, country/fiddle and jazz. The songs for those genres are Vivaldi’s Violin Concerto in A minor movement 1, Boil ‘em Cabbage Down, an American folk song, and Autumn Leaves by Joseph Korma, respectively. When a user clicks on a genre, the genre’s song starts playing and the violin animates and “plays” with the song.
4.3.2 Design and Implementation Choices

Figure 9: Block diagram of the hear the violin subsection architecture.

Figure 9 shows the architectural design of the hear the violin subsection architecture. There are two parts to depict realistic bow animation. First is the issue of lining up the animation with the audio file. I created the TrackSongPosition class that operated as the timekeeper of the scene. I then created the SyncedAnimation class that took the timekeeping variables of TrackSongPosition and the bow animation to play each frame of the animation exactly in line with the loop position of the song. Then the next problem that I solved was changing the bow’s angle to go onto another string in order to add realism. I extracted the note order and name from the MIDI version of the audio file and then I compared the current note and the next note to see if the bow needed to change strings. With these two components combined, the bow looks like it is realistically “playing” the song.

4.3.3 Animating the bow

For the hear the violin subsection, when I mention animation I will be talking about the bow of the violin animating. The bow moves back and forth in coordination with the song’s notes and changes strings when necessary. I created an animation for the bow in the Unity Animation feature seen in Figure 10.
Figure 10 shows the animation curve of the bow animation. The X axis represents time elapsed and the Y axis represents the coordinates. The bow’s position is shown as three curves that correspond to each of the coordinates; the bow’s X coordinate is red, the Y coordinate is green and the Z coordinate is blue. There is no change in the Y coordinate during the animation but the X and Z coordinates increase, reach a peak half a second in, and go back down to a similar coordinate. This change in the X and Z coordinates is shown in the Unity scene as the bow moving back and forth on the string.

Animations in Unity are created with keyframes which are the positions or rotations of a GameObject that you want to transition between. GameObject in Unity is a base class that is used to represent all objects in a scene. Unity automatically creates animation transitions between these two points, as seen by the curves connecting each diamond-shaped keyframe in Figure 10.

4.3.4 Lining up the audio file with the bow animation

Next, I needed to match up the bow animation to the audio file I wanted to animate with. I first attempted to play the animation at the tempo given by the audio file but I discovered that I had to measure time directly with the audio in order to accurately communicate between the
animation and audio. Lining up audio and video together is a very difficult task because humans are very sensitive to inconsistencies between the two. It ruins the experience if they are off at any point.

I tried tracking time with Unity’s Time.time measurement but it was based upon when the game started so if there was a lag during run-time, the timing would get off track from the audio file. I learned that I had to use Unity’s AudioSettings.dspTime, which according to Unity’s documentation is “based on the actual number of samples the audio system processes and is therefore much more precise than the time obtained via the Time.time property” [14].

In order to track the audio time throughout the game, I created a TrackSongPosition class that functioned as a timekeeper of the song position throughout gameplay. The TrackSongPosition class keeps track of the song’s Beats Per Minute (BPM), the number of seconds that each beat took, and the current song position in seconds and beats [15]. For each frame the song position is updated based on how many seconds has elapsed since the beginning of the song. There are also two other variables called LoopPositionInBeats and LoopPositionInAnalog that help assure that the bow animation is in line with the timing.

Next I could control the bow animation with the TrackSongPosition class tracking properties. I created a new class called SyncedAnimation that had the bow animation and an instance of the TrackSongPosition class as inputs. It started each frame of the bow animation at the corresponding loop position of the song so that the animation and audio would always be in sync.

With these changes, bow animation played in line with the song, so the animation and audio were synched together. This helped immensely with my goal of making the animation look as realistic as possible. However I still had to add one more element to the animation to make it look realistic to the user.

4.3.5 Changing strings

The next element to add to make the animation more realistic was changing the string that the bow was on to play the appropriate note. Rather than getting timing information from the song, I needed to extract information about all the notes in the song and their timing in order to know whether the bow needed to switch strings to play the next note.

In order to access the song and its notes I had to convert the audio recording to its MIDI version and then extract the MIDI file’s information. MIDI, short for Musical Instrument Digital Interface, is a way for computers to understand and play music. Computers cannot understand
written sheet music and musical note notation like humans can. Instead, they are given numerical information about each note in a song such as the note pitch, velocity (how loud or soft a note is) and the note’s starting and ending time. I used an open source .NET audio library called NAudio to extract the note information from the MIDI [16].

The way that the MIDI information extraction process normally works is that you loop through the whole song before game runtime to get all the information, then you can use the information for some purpose during runtime. However, I wanted to have the MIDI file loop through each note as it was played during runtime so that the audio file and MIDI file would be playing at the same time and I could switch strings at the right time. To loop through all the notes I used something called a Coroutine which is a loop in C# that delays a certain amount of time before continuing onto the next iteration. I wanted to delay the loop by the difference between the current note's duration in milliseconds and the previous note’s duration so it would wait for the duration of the note before moving on.

Below is the pseudocode to extract the note information from the MIDI file.

Pseudocode:
For each Event in MidiFile
  For each MidiEvent note in Event[i] up to Event.Count - 2
    Retrieve the NoteOnEvent of note and nextNote
    //NoteOnEvent is an object that contains a starting note’s pitch, velocity, etc.
    Convert note and nextNote’s AbsoluteTime from ticks to milliseconds and add to two separate lists.
    //formula: (60 * note.AbsoluteTime)/(bpm * deltaTicks). AbsoluteTime is the time that the note starts from the beginning of the song.
    delta = noteTime[note - noteTime[note-1]]
    Convert MIDI note to standard note notation
    //For example, A5 or G6
    Wait for delta seconds

After performing this process, I extracted all the notes, their durations and when they occurred. I could now use the information to change the bow angle to a new string if necessary. The process of changing strings has a few steps. I need to check if the current note and next note is the same string. If they are not on the same string I need to interpolate the rotation of the bow for the correct string transition (ie. from the A string to the D string).

I created two dictionaries in order to store a note and its corresponding string, and the string and its rotation angle, respectively. If the note and next note’s strings were not the same, I
passed it to another function that spherically interpolated, or Slerped, between the bow’s parent’s current rotation and the string rotation that we want to move to.

I interpolated the bow’s rotation through an empty parent GameObject because the interpolation needed to be performed in the world’s coordinates. The bow was moved with respect to the violin’s strings so if the bow’s rotation was directly altered it would change the local coordinates of the bow but other GameObjects would not recognize that there had been a rotation change.

After going through this process the bow changes automatically to the next note’s string before playing the next note. Then the bow animation continues with respect to the audio’s song position. With these two components meshed together, the bow looks like it is “playing” the song and changes strings when it is appropriate. It aids in making the experience look more realistic.

4.4 **Play the violin** subsection

The *play the violin* subsection simulates how one would play the violin. In the desktop version, it is shown as a note-playing game.

4.4.1 **Organization of play the violin**

![Block Diagram](image)

Figure 11: Block Diagram of the flow for the *play the violin* subsection
The *play the violin* subsection first asks if the user can read music. If not, it gives a quick tutorial of how to read sheet music, what the strings on the violin are called and all the notes necessary to play Twinkle Twinkle Little Star. Then there is a practice note game that leads the user through how to use the controls. Finally, there is an accuracy note game that is similar to the practice note game that is similar to a Guitar Hero format of rhythm games.

4.4.2 The Practice Note Game

First there is a section to orient the user with the controls as seen in Figure 12.

![Figure 12: Practice Note Game to orient the user with the controllers.](image)

In the desktop version, each note is mapped to the corresponding letter on the keyboard. For example, the note A maps to the letter A, the note B maps to the letter B, and so forth. There is a red bar that indicates the user’s current position in the game. The game is not timed and the red bar only proceeds to the next note when the user hits the correct letter on the keyboard.

The design and organization of this initial game is as follows. The whole game’s functionality is controlled by a GameManager script. It controls when the camera and the button moves when a note is correctly hit.
The game is designed like sheet music. All the notes and musical attributes like the treble clef, sharps and time signature (the 4/4 in Figure 12) are Sprites, which are 2D Graphic objects. The staff (the five long horizontal lines in Figure 12) are simple Quad GameObjects provided by Unity3D. The musical attributes are stored in the SongSetup GameObject and the staff is stored in the Staff GameObject in Figure 13.

![Figure 13: Organization of the practice note game in the Hierarchy panel in Unity 3D](image)

Next, all the notes are stored in a GameObject called NoteHolder, seen in Figure 13. Each note has a NoteObject script attached to it which has the following attributes mapped to it: the key on the desktop that is associated with the note, the AudioClip associated with the note and the AudioSource that will play the note. I broke up each note in separate audio clips that play when the correct note is played because it would be impossible to stop and start the whole song when we do not know when the user will press the correct note. In the NoteObject script, when the correct key is pressed, the AudioSource is given the AudioClip to play, and the GameManager moves the red block to the next note and the camera to the center of the next note.

The first move function that the GameManager controls is moving the red block to the next note. At runtime, the GameManager class creates an array of all the Transforms (ie. positions) of each note, called noteList. The move function is passed the note that was pressed. It finds the index in noteList of the note, calculates the new position it should be at by getting the X coordinate of noteList[index+1], and then performs a Slerp transform to move the red block to the next note’s position.

The function to move the camera’s position is simpler than moving the red block. Each note is positioned three blocks away from each other so I just increment the current camera position’s X coordinate by three.

After the user finishes the whole game, they are given the option to repeat the game to get more practice, continue onto the accuracy-based note game or go back to the home menu, as seen
in Figure 14.

![End Panel after the user has finished the Practice Note Game.](image)

**Figure 14:** End Panel after the user has finished the Practice Note Game.

### 4.4.3 Accuracy-based Note Game

The accuracy-based note game looks almost identical to the user as the practice note game, in Figure 15. However the functionality is more complex. In this game the user must remember the note names and which key to hit. There are also different scores depending on where the note is pressed inside the box. Additionally, the notes move towards the stationary red block.
The AccuracyNoteHolder, the GameObject that holds all the notes, has an AccuracyBeatScroller script attached to it. This script moves all the notes at a speed relative to a given tempo. When a button is pressed in order to start the game, as seen in Figure 15, the AccuracyNoteHolder’s position is decreased in the X coordinate for each frame update. The formula is the BPM of the song multiplied by the time elapsed from the last frame in the game. This allows all the notes to move smoothly at the speed needed to stay in time with the song.

Each note has an AccuracyNoteObject script attached to it. The key to press is mapped to the note like in the practice note game however there are no AudioClips mapped to the note. Since it is accuracy based, I lined up the notes with the whole song so I did not need to break it up into individual notes.

Inside the AccuracyNoteObject, when the correct key is pressed for the note, the note is disabled (ie. disappears from the screen) and it is calculated whether the hit was Normal, Good, Perfect or a Miss. A Normal hit is when the note was touching the red box, either when it came in or as it was going out. A Good hit is when the note was completely inside the red block but not hit in the center. A Perfect hit is if the note was hit in the exact center of the block. For each calculation, the AccuracyGameManager NormalHit(), GoodHit() or PerfectHit() function is called to add the corresponding points to the total score shown on the screen and keep track of
the types of hits that were made. It also keeps track of and adds multipliers if there have been many good hits in a row. A Hit, Good, Perfect or Miss text is also instantiated that will display over the red block.

After the game is over, a Results Panel will pop up with the number of Normal, Good, Perfect or Missed hits, percentage of accurate hits, end score, and a final ranking of Amazing!, Good, Bad and Awful, as seen in Figure 16.

Figure 16: An example Results Panel after playing the accuracy-based note game.

5. Results and Analysis

My initial goal was to have a high quality prototype with three fully-functioning subsections; parts of the violin, hear the violin and play the violin. I was able to fully implement the parts of the violin subsection, including the initial tutorial and the extended tutorial. I learned a lot about how to make UI elements abstract - meaning using the same Menu over and over again and populating it with new data. This helped with decreasing the space complexity and clutter of the whole app.

The hear the violin subsection turned out to be much more complicated than I anticipated due to the challenge of lining up the violin’s bow animation with the audio file. Ultimately I was able to animate it without any issue if the notes were the same length, for example, quarter notes. I did attempt and got to a point where I was able to animate the bow when there were different
lengths of notes - for example, eighth notes and sixteenth notes - but I had an issue where the bow would stop after it changed to a faster note. For example, the bow would initially “play” eighth notes correctly and switch to sixteenth notes but whenever the notes went back to eighth notes, the bow would freeze up and only animate again when it had sixteenth notes. I was also able to have the bow switch strings correctly using the MIDI extraction technique described in 4.3.4.

I was able to fully implement the play the violin functionalities for the desktop version. This included a practice note game to get used to the controls on the desktop and then an accuracy rhythm game using the Twinkle Twinkle Little Star song.

6. Future Work

If I had two to three more months to dedicate to this project, I would port the desktop version to make it VR compatible as well. I would also create the VR components for the play the violin subsection. For that section I would need to build a tutorial to go over how the Oculus Rift controllers work with this application and also map the controller buttons to certain functions to simulate playing the violin. The left controller would represent the fingers pressing the notes on the violin and the right controller would represent the bow. On the left controller, no buttons pressed would represent the open string, the first button pressed would represent the first finger, second button for second finger and third button for third finger. For the right controller the user would move their arm back and forth at specific angles to simulate playing on a string. The closer their elbow is to their torso, the higher the string will be and vice versa. I would need to program this mapping to the Oculus Rift controllers.

I would also fix the hear the violin algorithm so that the violin can animate correctly with any type of note and switch between them. I would also like to make it possible to abstract the hear the violin algorithm to work on any song.

After the main functionalities of the app were tested, I would release it to beta testers - ideally elementary-school aged children, who are my target audience - and they could give me great advice on how to improve the app. While testing the app with children, I would also like to investigate whether using VR for music education actually improves children’s effectiveness of contextual learning and improves their understanding compared to current teaching methods. As I mentioned in the Introduction, it has been shown that learning through VR allows children to make more contextual connections to their material and therefore leads to better grades and test scores. It has been proven to work in academic settings, but it is harder to prove its effectiveness in a music education setting. While testing I would like to learn if that is true and if the benefits of VR education in academic subjects also apply to music education.
If I had six more months to dedicate to this project I would completely redesign the UI of the app. I was using very basic UI components provided by Unity3D and it looks very bare and not entertaining. I would design a color scheme and layout that would be appropriate for desktop and VR. I would also add more VR UI components and flourishes to make it a more exciting experience. Lastly, I would add quizzes and/or minigames to the violin sections so that the user can map their progress and the app can be more gamified.

If I had a whole year to work on this, after I complete all the parts mentioned above, I would venture out and apply what I built with the violin to other instruments. My ultimate goal is to create this app for every instrument in the orchestra, thirteen in total, so I would work to add more instruments. Since the majority of the effort was in the initial design and programming of the violin, I believe making additional sections, at least with string instruments, would be much easier and the amount of editing and customization would be minimal.

7. Summary

Primary and secondary schools’ arts budgets have continually been cut because of their status as non-essential subjects [1]. However, music education has been proven to be extremely beneficial for child development as it fosters discipline, increases creativity and memory capacity and contributes to higher performance in academics [2]. This project seeks to give children a comprehensive personalized music education experience while cutting costs for school districts who may not have the funds to buy instruments and refurbish them. The VR Orchestra App provides music teachers a supplemental tool to teach about instruments in orchestras. After using the app, children will be able to identify instruments visually and aurally and remember core characteristics about each instrument and about the orchestra as a whole. This project built a fully-functioning prototype of the violin section for desktop use. In the future, the app will be developed for the Oculus Rift and will be fully implemented for all thirteen instruments in the orchestra and include quizzes and mini-games.
References