Computer Science Project, 2017-2018
Luleå university of technology

VR MusicBox
Final Report

Léandre Le Polles-Potin
Enzo Menegaldo

Abstract

The project “Music Box VR” is an application which aim to allow several users to interact in the same virtual reality environment to play and create music together. The application supports the use of either HTC Vive and the GoogleVR Daydream technologies. This report will describe the different trails explored to meet to aforementioned objective as well as the final result of our work.

The project is available on GitHub at the following address: `www.github.com/LeandreLPP/VR-Music-Box`

Contents

Introduction

This project took place in the context of the "Computer Science Project" course at the Luleå University of Technology, and is supervised by Professor Peter Parnes.

The first goal of this project was to explore the use of Virtual Reality to create an interactive music application where people out of the VR environment could interact with the user inside. The main idea was to use HTC Vive Trackers, manipulated by users outside the VR to allow them to interact with someone using the VR headset.

Our first idea was to allow the creation of music using sound loops. The users could change the tempo and the pitch of each loop by moving the associated HTC Tracker in the room. However, we noticed that the usefulness of a such application was limited.

Then, we decided to try to use the trackers associated to a smartphone to create a second headset in the VR environment. This could have allowed several users to interact in the same environment with just one HTC Vive and at least one tracker per smartphone.

Finally, after reaching that goal and analysing our conclusions, we decided to give up with the trackers and to turn towards a new technology: GoogleVR Daydream. The main idea was to allow several users, either with an HTC Vive or with a smartphone able to support Daydream, to interact in the same environment to play and create music together. The application uses the principles of a sequencer to achieve that.

The project was actually very open, and the real goal was to explore multiple subject around VR, music creation, and the way those two subjects could merge.

Technology used

2.1 HTC Vive

During this project, we mainly used an HTC Vive set containing one virtual reality headset and two controllers. The HTC Vive is a high end virtual reality headset developed by HTC and Valve Corporation. The headset uses "room scale" tracking technology, allowing the user to move in 3D space and use motion-tracked handheld controllers to interact with the environment. Additionally, we used some HTC Trackers. These devices allow to bring physical objects into virtual reality by fixing them on the objects. They work like the handheld controllers, minus the buttons.

Figure 2.1: HTC Vive

Figure 2.2: HTC tracker

2.2 TP Cast

The HTC Vive uses a long cable that does not appear in the virtual world to connect the headset with the computer. The TP-Cast system allows the headset to work wirelessly with little to no lag. We used it during the entire project without encountering any major problems. However, some online review pointed out a decreased field of view while using the TP-Cast system.

2.3 Google VR

We used the Google VR technologies to bring this application on smartphones. We started using one Google VR Cardboard (Figure 2.4) and our personal smartphones then using a dedicated Google Pixel (more powerful) and Google VR Daydream which is the more powerful VR platform for Android devices. Normally, we must use Daydream with a dedicated headset (Figure 2.3) which contains a controller. We can't run a Daydream application without this controller. Unfortunately, we didn't have one. Fortunately, Google VR has created a controller emulator which can be run on another smartphone.

In brief, we have used a Google Pixel to run the VR application, a typical smartphone to run the controller emulator (Figure 5.1) and a Google cardboard to visualize the VR application with the Pixel.

Figure 2.3: Daydream Headset

Figure 2.4: Google Cardboard

2.4 Unity and Photon PUN

During the whole project, we used Unity to create our application. For the networking, we have used the framework Photon PUN which is a framework for real-time multiplayer games and applications developed with Unity. We decided to use it because it ensures very good performance and reliability with an easy deployement.

First explorations

Our main goal was to create an interactive music application where people, who are outside the VR, can interact with a user in the VR environment.

The main idea was to use the new HTC Vive trackers, which being manipulated by the users outside the VR could allow them to play with someone in the VR application.

Besides, another purpose of this project was to evaluate the usability and usefulness of multiple HTC Trackers in a VR application. To do so, we tried using these trackers in several ways.

Our first idea was to allow the creation of music using sound loops associated to trackers. The users could change the tempo and the pitch of each loop by moving the associated HTC Tracker in the room.

Our second idea was to use trackers like instruments. By moving the device in the right way and in the right tempo, the users could have been able to play music.

3.1 Sound loop receptacles

This was the first use developed for trackers in this application. The basic idea was to tie a sound loop to the trackers and use the position of the tracker to change the tempo and the pitch of the sound playing.

Doing so proved to be easier than expected, the Steam VR asset package providing an easy way to integrate HTC Trackers into the application.

However, the way some flaws in the positioning made the tracker "jump", requiring some adaptations. The way we tried accounting for this "twitchy" behaviour was by making changes to the pitch and the tempo by step rather than linearly. This allowed us to disregard some of the most minor imprecisions in the position detection. Despite this, when nearing the border between one step and the next, the unwanted behaviour would appear again. This require the user to move the tracker away from the border one way or the other.

Other ideas were explored, such as tying the position of the object in the VR environment to the moving average of the recent positions of the tracker. However, this complexified the code structure and the object structure in the Unity scene for too little gain of reliability. This method could still prove useful if the precision and stability of the trackers was a more important factor.

Our goal reached, we faced the disappointing conclusion that the Tracker didn't bring much, the usefulness of a such a device for this application was limited. The same result could have been achieved using a simple virtual object grabbed and moved using classical HTC controllers.

3.2 Use trackers like instruments

The second idea was to use the trackers like instruments. Of course, we didn't think about using them to play complex instruments like guitar, violin, piano, etc. But basically, we could have used them to play some little percussion instruments like maracas, claves or bells. This kind of instrument just has one sound, so it seems easy to simulate their behaviour with only one tracker.

Therefore, we tried to create maracas with a tracker. We made this choice because it was the most complicated one. When you play maracas, you need a regular rhythm at high frequency, so you need a good reliability at the detection of the movement.

It allowed us to find out the limits of the Trackers for this kind of use. Once again Steam VR directly provided the methods needed, in this case the velocity of the tracked object. We simply played a sound each time a significant shift in the direction or the velocity was observed. Nevertheless, the expected behaviour was not reliable enough for our use.

Here we are tracking a constant movement and not just a position like for our first idea. Even if this was a more interesting and engaging result, the use of a Tracker felt forced, as the same code could have been applied to any virtual object or a basic controller.

Building environment and first networking

In the meantime, we developed an environment in which the user could play and create music. The main goal was just to create a pleasant setting so there were no real technical challenges at first, but we also tried including some multi-user experience, which proved a bit more difficult.

4.1 Instruments

After finding out that simply assigning sound loops to objects in order to change their pitch and tempo was not as powerful and interesting that we once thought, we started designing other ways to create music in the virtual world. We began by creating three instruments. A vibraphone keyboard, a drum set and a harp-cello (Figure 4.1).

The vibraphone is simply a collection of tiles one has to hit in order to produce the correct sound. At first, any collision would play the sound, but using the HTC controllers to play a melody proved too imprecise for our taste. We created sticks, which have a smaller hitbox and are more intuitive to use but were confronted to another problem. We couldn't make the sticks react to the collisions with the tiles realistically because that would have required to give a physical feedback to the user. At the same time, having sticks pass through the tiles meant that lifting them would play the sound another time. Finally, we settled for a simple check of the direction the stick went when colliding with the tiles, which proved to be the most intuitive way to simulate the playing of a real vibraphone in VR.

We used the same technique and principles to play with the drums set, while the harp-cello simply starts playing a sound when a string is touched and stops playing it when the string is released.

4.2 Environment

In order to make the virtual world more visually appealing, we designed an environment with the assets issued from the free asset package *LowPoly Environment Pack*. However, we didn't want to focus too much on the world building, because it proved to be a time-consuming task for little to no technical gain.

That meant that the user was constricted to a small play area, from where to world appears alive and full. Allowing the user to move around the scene would have quickly revealed a much poorer scenery. In result, we had a much more visually pleasant, but much more restricted environment (Figure 4.1).

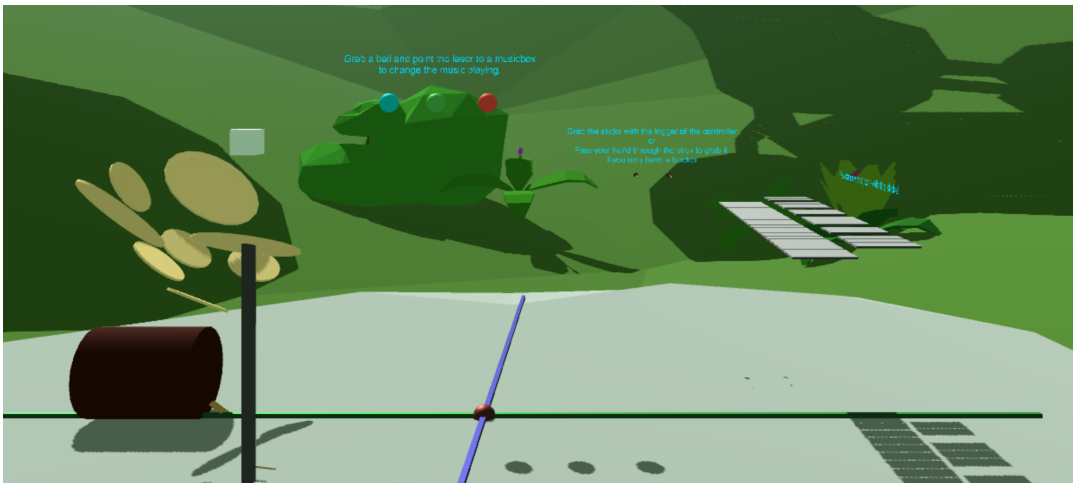

Figure 4.1: First environment

4.3 Localization of another screen

Once our environment built, we focused on another aspect we wished to explore: the interaction between multiple users in and out of the virtual reality.

The idea was to connect a smartphone or tablet to the main application using Unity's networking tools and track the device's position using an HTC Tracker. For this use, the Tracker proved to be an excellent tool, reactive and precise enough to transform any Android device into a VR camera. Splitting

the vision in two even allowed us to use the mobile phone as an improvised VR helmet, using a Google VR Cardboard. It would have required a custom-made case attaching the device and the Tracker together for the solution to be perfect, but this is a minor and easily corrected annoyance.

The most difficult and time-consuming aspect of this solution was to synchronise everything between the two applications. We decided to go for an asymmetrical structure, the desktop connected to the HTC Vive playing the role of the server, while the mobile phone or tablet played the role of clients. This allowed us to keep using the high-level networking API of Unity to obtain quick results. We managed to create this second application and the smartphone acted like a real VR headset.

To allow the external user to interact with the VR world, we configured a tracker to act as a makeshift controller. This proved to be one of the best uses of the HTC Tracker that we found. Thanks to that Tracker, two users could truly interact within the same VR world using an HTC Vive and two HTC Trackers.

However, we could observe some lags between the two applications and that brought a major problem on what we wanted to do next.

Our application was a set of instrument. The main idea was to allow several users to play together, one with the HTC Vive playing vibraphone and another with the trackers playing drums for instance. However, the synchronisation between the users is practically impossible. Even if there was no delay between the applications, playing music with other peoples is already difficult. With a variable delay, created by the lag between the applications, our idea falls apart. The reliability, the efficiency and the utility of the system were too low so we decided to stop there and to turn to another concept of application.

Focusing on multiplayer

After testing to use a tracker fixed to a smartphone like a second headset in the application. We decided to give up the use of the trackers and to turn towards the Daydream platform developed by Google VR. Daydream is the more powerful VR platform on Android. Our idea was to use this technology to improve the experience with the smartphone. In addition, to fix the issues of latency, we decided to implement a music sequencer to create and play music.

5.1 Google VR

Google VR Daydream allows us to import our application on Android smartphones. We can use several HTC Vive to play together in our application. However, it's not common to have an HTC Vive and even more unusual to have several HTC Vive in the same place. Besides, each HTC Vive require a powerful computer. So, if you want to use this application as a multiplayer game, it would require a lot of expensive hardware.

The goal was to facilitate the use of the application by enabling its utilisation on smartphones which are much more common. Of course, the experience will not be as immersive as with the use of an HTC Vive. However, Google VR Daydream provides a controller which allow the user to do the same things that the HTC Vive user but in a different way. By using Google VR, the smartphone only detects the rotation of the device, not its position, so we can't move in the virtual space like with an HTC Vive. But thanks to the new controller, we have a laser. This allows the users to interact with the virtual environment. For instance, they can grab and release objects by clicking on the touchpad (Figure 5.1) or they can move in the space by using the teleportation associated with the laser by clicking on the App button (immediately below touchpad, Figure 5.1)

5.2 The Music Sequencer

Our music sequencer is made of one central column of receptacles and a circular wall of switches (Figure 5.2). The wall is divided into rows and

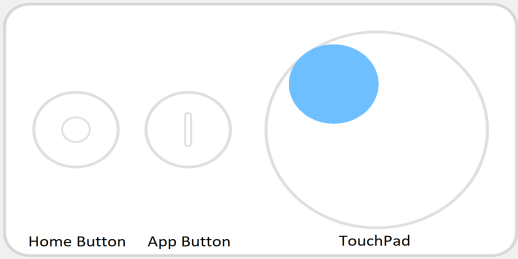

Figure 5.1: Google Controller Emulator for smartphone

columns. Each row is associated to a receptacle and each column represents a step. The sequencer plays successively plays the steps at a regular tempo.

If the user wants to use a note, he has to put the desired note in one of the receptacles. Then, the user has to toggle on the same row, at the desired step, the associated switch. When the sequencer will reach the step, it will play all the notes in the central column which have their switch toggled on at the step.

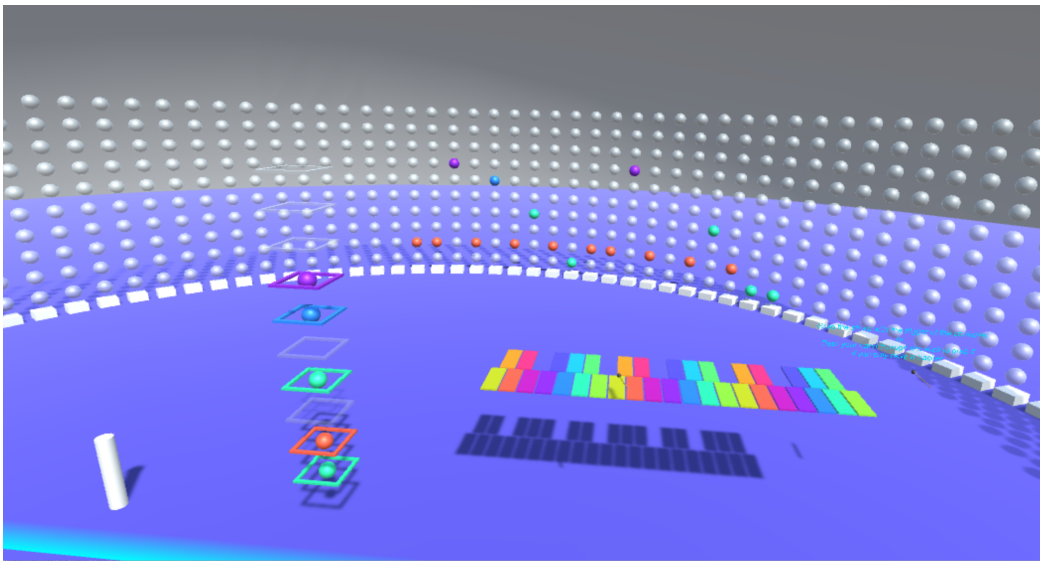

Figure 5.2: Sequencer

5.3 The notes

5.3.1 How to create notes

In the application, a note is represented either by a sphere or by a cube. Currently, a vibraphone (Figure 5.3) is put at the disposal of the user to create notes.

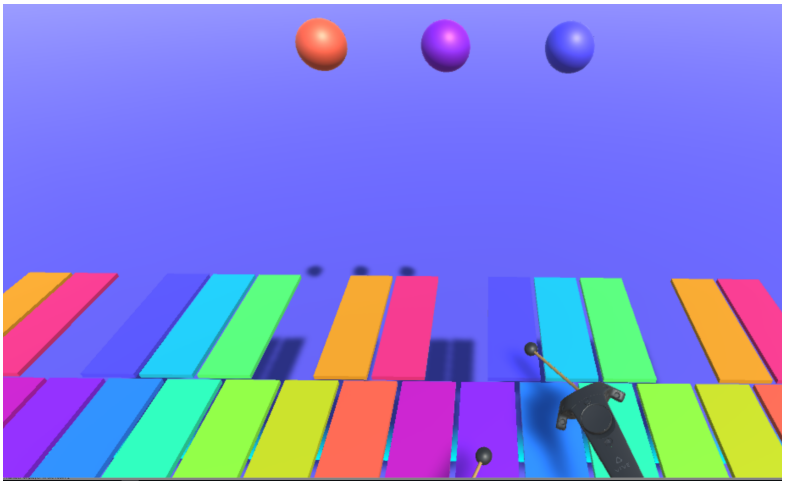

Figure 5.3: Sticks and Vibraphone

If a user plays the vibraphone, a note will be created in all the instances of the game (i.e. for each users). These kind of note will be spheres. The user is also provided with an user interface, which allows him to select a note to create (Figure 5.4).

These notes will be represented as cubes. Currently, the user can only select sounds from drums.

Playing vibraphone is different with each technology used. If the user uses an HTC Vive, he will have to pick up a stick (Figure 5.3) with the controllers and hit the vibraphone with it. It will automatically instantiate a note with the desired sound. If the user uses a smartphone with Google VR, he will have to point the desired note with the laser and click on the touchpad of the controller to create the note.

If the users want to use drums notes, they must point their controller towards the user interface and to click on the button “play”. They can switch

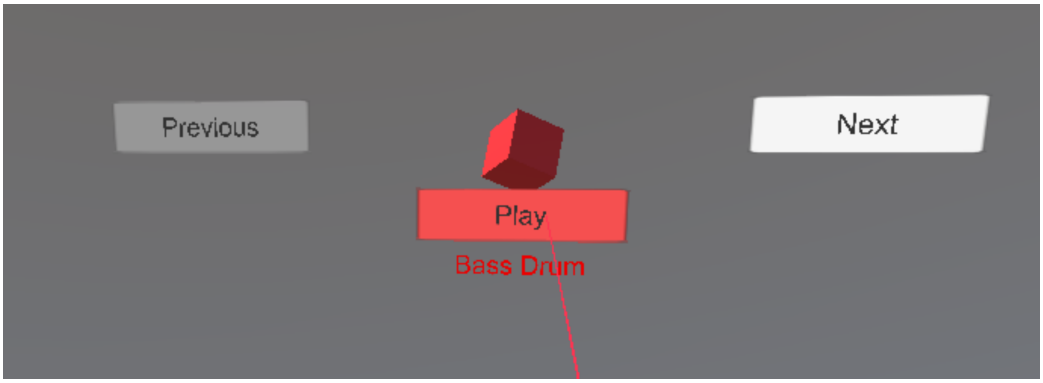

Figure 5.4: User Interface

the sounds by clicking on the buttons “previous” and “next”. (Figure 5.4). When the button “play” is selected, a note with a cube shape will spawn (Figure 5.5).

Figure 5.5: Notes spawner

5.3.2 How to interact with the environment

Now that the notes are created, we have to interact with them to put them in the sequencer. The interaction differs depending on the used technology.

If the user uses an HTC Vive, he can grab a note just by approaching one

of his controller and pressing the controller's trigger. He can add a note to the sequencer by releasing his held note in one of the central receptacles or throwing the note toward the receptacle. To toggle a switch, he just has to pass through it. The application simply uses a collision detection between the controllers and the switches.

Moreover, the user can move in the environment, of course by moving in the real space but also by using the teleportation. For this, he just has to press his controller's touchpad to reveal a laser, to point with the laser to where he wants to go and then to release the touchpad.

If the user uses a smartphone, it's a bit different. We cannot use the collision detection, so we will only use the laser to interact. The user can grab a note by pointing it with the controller's laser and clicking on the touchpad. He can release it by pointing the floor and clicking on the touchpad. If he wants to add a note the sequencer, first, he has to grab a note and then he must point the desired receptacle and click on the touchpad. He can take it back with the same manipulation. To toggle a switch on the sequencer wall, he just has to target the switch and click on the touchpad. The user can also move in the environment using the teleportation by pointing the floor and pressing the App Button on the controller (Figure 5.1).

Finally, the user has access to a user interface (Figure 5.6) to manage the tempo, clear the sequencer and run a demo. If the user uses an HTC Vive, he can also switch to the first forest scene. He will be able to come back in this scene thanks to another UI in the first scene. The way to use this UI is the same for both technologies: point the laser toward a button and click on the trigger for the HTC Controller or the touchpad for the Daydream Controller.

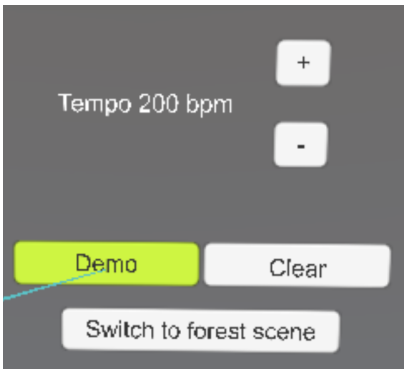

Figure 5.6: demo and tempo UI

5.4 Tempo controller for sequencer

Finally, we decided to add one tracker in our application. The goal was to tie the rotation of the Tracker to the tempo of a sequencer. So, anyone outside the application could interact with it by rotating this tracker. The tempo of the sequencer is the same on each instance of the game. Therefore, if someone rotate with tracker, the tempo will change for every users.

We managed to do it without much problems, but with the same conclusions than previously: the utility is limited and the use of the Tracker feels forced.

5.5 Networking

Using a sequencer allows us to struggle against the latency. In our application, the notes, the switches and the tempo are networked. So, every instance of the sequencer plays the same music with the same notes at the same rhythm. However, the sequencers are not synchronized between the instances so there is a delay between each instance. This is not a problem because the delay is constant. For instance, an user can be 50 steps behind the first user (the one who was the first to launch the application) and will always stay 50 steps behind him. That way, even if the users don't hear the music simultaneously, they all hear the same music.

A user can interact with all the notes, even those he didn't create. When a user moves a note, the others see it moving in their instance. The only thing a user can't do, it's to grab a note which is already grabbed by another user.

If a user changes the tempo, it will change in all the instances of the game.

If a user toggles a switch, it will be toggled in all the instance of the game. This allows us to ensure that on every client, the music is synchronised.

To realise the networking of the game, we used the framework Photon Pun. It is a framework used to develop real-time multiplayer games and applications on Unity. We decided to use it because it ensures very good performance and reliability with very little to do on our end. Actually, the application uses the server of Photon, so the application can be used in different networks. We could have set up a local server but Photon's server being reliable and efficient, we decided to keep this configuration.

The way we use Photon is quite simple. Each note belongs to a player. When a player create a note by playing vibraphone or using the UI, he will get the ownership of the note. When a player grab a note, the ownership of said note will be transferred to him.

Each client sends to the others the position of the notes he owns. The connection between the clients is unreliable so there can be some losses. Yet, the data are sent fast enough for the application to synchronise the position between the clients. Moreover, the data are sent constantly so even if we get some losses, it's not a problem because the client will resend the position in the near future.

The other events (toggle a switch and change the tempo) are managed in a different way. When one of them happens, the client will send a RPC call to all the other clients to notify them to update their instance.

The RPC calls are not buffered. It means that when a player joins the game, he will not be aware of what happen previously except for the creation of the notes and their position (it's managed by Photon).

To synchronise the switches, the sound in the notes, the tempo, etc., the application uses the principle of "Masterclient". A Masterclient is basically a client which has more responsibilities. When a player joins the game, all the current players are notified. Thus, when a player joins, the MasterClient will send him the state of all the current objects in the game.

Learnings

Thanks to its exploratory nature, this project allowed us to touch and learn many things about Unity and the development of Virtual Reality applications. The technical aspects of working with virtual reality was surprisingly easy, thanks to the ready-to-use tools provided by the asset package SteamVR. However, having the opportunity to use such a device allowed us to better comprehend the possibilities and limitations of Virtual Reality.

6.1 Unity

First of all comes Unity. We both had the occasion to use Unity before, but this project deepened and improved the way we understand the structure of a Unity project. Amongst many other things, we can cite the use of Vectors and Quaternions to place objects in the scene using scripts, the versatility of raycasting, the basics of Unity Networking API and the importation of asset packages to aid us.

Rather than focusing on our mastery of the technical tool, Unity, we will try to explain the principles of VR development that we learned about.

6.2 GoogleVR Daydream

Much like SteamVR for the HTC Vive, Google provides an asset package allowing Unity developer to create VR application for Daydream-ready mobile phones.

This solution is far more limited than the HTC Vive, but allow a wider public to enjoy VR applications because of the reduced cost.

The way VR is implemented by the Daydream asset package is different than with SteamVR. Both link the position of the headset and the controller in the real world to objects in the virtual world. However, while SteamVR centralise the management of inputs to these virtual object, Daydream also allows the creation of listeners on every interactable object in the scene. Listeners will be called to the object when the user point the controller

towards it and presses a button.

This allows a better flexibility and structure in the scene, dividing the logic in several easy-to-fix blocks rather than grouping everything in a single script file. However, using multitude of hardware-specific scripts is not a good idea for a multi-platform applications. For such applications, we recommend keeping a centralised input receiver for each hardware and calling non-hardware-specific methods to interact with objets in the scene.

6.3 Movement in VR world

Early in the project, we faced the problem of movement, one of the main problem faced by VR currently. While developing VR applications, one have to keep in mind that the user is limited to the size of his room to wander in the VR world. With some VR hardware, the user can only rotate and possess no liberty of movement. To expand the playable area, the developer have to think of ways to make the user move in the virtual world without him actually moving in the real world. However, a continuous movement in the virtual world tend to make the user sick or nauseous because of the disparity between the visual stimuli and the movement (or lack of movement) perceived by the inner ear.

The simplest way to work around this problem is to allow the user to teleport, getting rid of any visual representation of the movement and the associated sickness. The user simply point a controller toward the place on the ground and press a button to instantaneously move at his location. This method was the one implemented in our application because of its simplicity, but it is interesting to mention some alternatives that we learned about.

The first alternative we discovered was implemented in the VR game *Waltz of the Wizard* and is called *Telepath*. Using a controller, the user traces a path on the ground. Once traced, the character in the virtual world will periodically hop to regularly placed point along this path, essentially trading a single long teleportation with a multitude of little ones. In the context of this game, this solution allows a better interaction with the non-player characters that could get easily avoided by the mean of teleportation.

Another way to deal with the dissociation between the perceived movement and the visual stimuli is to force the user to move in the real world in order to move in the virtual world. For example, leaning in one direction cause the

character to move or dash in the same direction. Many teams works with variation of this principle, such as the *MonkeyMedia* tech firm in Texas.

Giving the user a frame of reference, such as the dashboard of a cockpit or a car, can also help reduce the motion sickness. This method tend to give mixed results and cannot be considered sufficient to fully get rid of any kind of motion sickness.

With these options in mind, one can design the movement system that one wishes without causing motion sickness for the user. It is also interesting to mention that everyone is not equally sensitive to motion sickness.

6.4 HTC Trackers

The use of HTC Trackers in Unity proved trivial with the Steam VR asset package. The main problem we encountered was to connect multiple HTC Trackers to the computer. While in theory a computer could connect to as many HTC Tracker as it has available USB slots, we never managed to pair more than three Trackers at the same time, because of seemingly random pairing problems.

Another problem to account for is the fact that on the start of the application, Unity pair the physical Tracker with their corresponding object in the scene in a seemingly random way. That means that making sure each tracker is associated with the correct in-game object require the involvement of the user. This is not always advisable but inevitable unless all trackers used in the application are interchangeable.

Despite these limitations, the use and programming of HTC Trackers is still pretty simple and intuitive. One can easily tie the position of a virtual object to the tracker and do whatever one wants from this point on.

However, in the context of this project, finding a truly engaging and interesting use for them proved difficult. We were satisfied by the way the Trackers reacted and we considered our goals reached but, the final result was disappointing as we found their use cumbersome and with no real added value.

While in the virtual reality, distinguishing what is purely virtual from what is the reflection of a real object like a Tracker can prove difficult. Added to this was the fact that the trackers were supposed to be left in place once

we were done using them (either to change the tempo, the pitch or to play some maracas). This forced us either to stay immobile with the Trackers or to put it on the ground, where it could be easily stepped on. Seeing that, we concluded that the use of the Tracker felt redundant and non-necessary, since a purely virtual object would have had given the same results without much disadvantages. Trackers should not be used for the sake of using them but to truly make real world interact with the virtual world, and we failed to apply this principle during this project.

This conclusion is not one saying that Trackers are useless, far from it. But one must remember that they are supposed to be used to track the position of real objects in order to make reality and virtual reality interact. Using them to track the position of object that were supposed to be immobile most of the time was not a good decision from our part.

6.5 Networking and multi-user VR application

The Networking and the exportation of the application on Android devices is a real success in our project. We manage to create a totally multiplayer environment where people can interact and create music together. Open the application to Android devices allows it to be more interesting, funny and accessible. Even if this application can be used only on the last devices which can support Daydream, it opens the game to a wider public.

Currently, the application doesn't possess a wide range of notes and instruments. However, as we developed this game trying to follow good architecture principles, it's now very simple to add new sounds and new instruments in.

The same applies to the use of the Photon framework for the networking: we can ensure the stability and the reliability of the application should the next additions follow the same structure.

Conclusion

This project allowed us to learn the development of VR applications using Unity. Since we focused on exploring different aspects and technologies, the application we developed is more a technical demo than a real finished product. However, we were careful to keep a correct structure in our development in order to give the tools for continuing this work to whoever is interested.

Thanks to its exploratory orientation, this project proved to be a real learning experience. We especially gained a lot of confidence in the use of Unity, the Steam VR and Google VR asset packages, which could be useful in the future.