Orchestra VR

New Technologies - Report 1

Erik Oussoren
Ananya Ghosh
Pepijn Thijssens
Felix Buenen

Project Summary

In this project, we have attempted to create a mechanic that could be implemented in an orchestra conductor experience. The player decides what instruments he wants to be played (by pointing and clicking) and at what tempo he wants those instruments to be played (by swinging his other hand in the desired tempo).

Motivation

Conceptual
We were interested in giving the audience the experience of conducting an orchestral environment with fun components and making music. While the goal of the experience is to cheer up the audience sitting behind you by making interesting music but if you go wrong, you can expect a few tomatoes thrown at you from behind!

Technical
For this project, we used the Vive VR system. We had a couple of reasons for this:

- By making the game VR, we could actually place the user inside the game, which would enhance the conducting experience.
- The Vive is known for being very accurate, arguably more accurate than the Oculus VR. This is very important for our game, since, as you will read in a moment, tracking player movement to measure music tempo is a tricky business which requires attention to the smallest player movements.

Decisions & Choices

Conceptual
Swinging around your arms while walking around on a small podium did seem like a perfect fit for the Vive VR radius and accurate controller readings. Some sound would make the game feel really gimmicky and might be a fun little demo to show off what a Vive can do.

We wanted to add another layer to the VR game where you have to dodge thrown tomatoes at you. However we got quite stuck with the BPM part of the game we never got the time to

implement some kind of throwing and hit detection on your body. Things as implementing legs and a body to your VR self which would be painted by the tomatoes was the original plan but never got to do it.

Technical

We soon realised that measuring the BPM (Beats Per Minute) based on player input was going to be a challenge. The player indicates BPM by swinging his controller in the tempo that he wants. If the player swing comes to a stop, we measure BPM as follows:

```
BPM = 60.0 / (currentHitTime - prevHitTime)
```

Where `(currentHitTime - prevHitTime)` is the time it took for two consecutive 'swing stops' (i.e. beats) to follow each other up. We then use the newly calculated BPM to change the pitch (and therefore speed) of the audio files, which can be achieved using built in Unity components.

The main challenge, however, lies in measuring when a player *starts* a swing and when a player *stops* a swing. Of course, we could say: "a player stops a swing when the measured speed of the controller is 0". But that never happens; players always move their hands a bit, especially when their hands come to a sudden stop (which is what usually happens after a swing). The trick therefore lies in finding the speed threshold, both for deciding when the player stops a swing and when the player starts a new swing (e.g. at what speed do we classify the controller as being 'still' or 'moving').

Although this sounds very easy, after lots of experimenting, we still haven't found the right thresholds. We tried to log our movement speeds in several ways, attempting to find the speed values for stopping a wave, but this has not been proven to be very effective.

A possible solution is to start with a configuration phase, where players swing their controllers based on instructions on screen. The algorithm can then try to build a small AI system which it can use to classify 'stops' and 'starts'. This would be beneficial in two ways:

- It would hopefully solve the problem of manually finding the thresholds.
- Different players have different swinging behaviours, and therefore also different thresholds. When a new player wants to play, he/she simply has to redo the configuration phase and will be good to go.

Due to time restrictions, we have not been able to implement this. Furthermore, we do see possible problems with this approach:

- Not everyone has a perfect sense of rhythm. Our algorithm would measure 'stop speeds' based on a configuration beat which the player has to follow. At every stop-event of this beat, the algorithm would measure the current speed of the player controller. If someone has a very bad sense of rhythm, he might still be swinging his controller, providing the algorithm would false input.
- A player's swinging behaviour might change from the configuration phase during the game. We could try to implement an AI that adapts intelligently to this, but that would make the algorithm a 'guessing game', which is not desirable with such sensitive

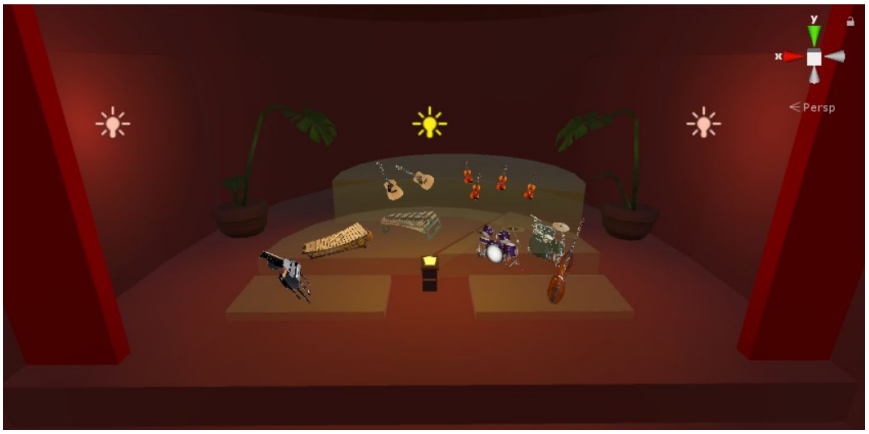
thresholds. What could work is to, during the configuration phase, ask the player to do different kinds of swinging. For example: swing the controller mildly, following a 120 BPM beat. And: swing the controller aggressively, following a 200 BPM beat. Maybe the player doesn't even have to follow any BPM at all and the algorithm can just intelligently identify 'stop' events. The AI could build a small neural network and classify movement accordingly.

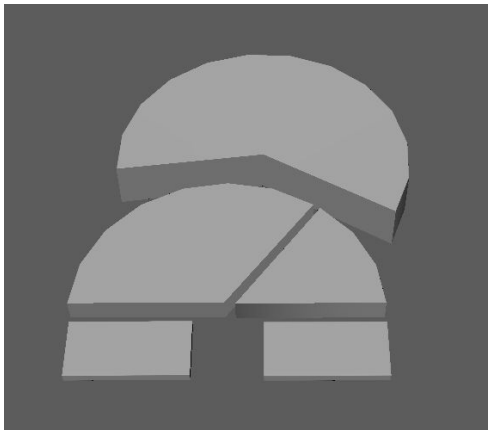
For the time being, we have created a small 'hack', that checks if the newly calculated BPM is higher than 400. Since this is extremely high, we say that such a high BPM means that the algorithm failed and the algorithm acts on it by not updating the BPM of the song.

In this case it was very easy to add some simple art to the scene to give the idea of a concert hall. To save time, we used pngs of instruments to show easily and quickly what you are toggling on and/or off.

Gallery

You will find a video demonstration of the game through the following link:
https://drive.google.com/file/d/1Zlri1tXfjCDMnIoZtDnvPZUg4m2j1bU3/view?usp=sharing





Team contributions

Who?	**What?**
Erik Oussoren	● Creating the scene ● Art Assets
Ananya Ghosh	● Orchestra Models - Maya ● Final Presentation ● Documentation
Pepijn Thijssens	● Programming ● Documentation ● Testing/Debugging
Felix Buenen	● Programming, BPM algorithm ● Testing / tweaking ‘swing’ values ● Documentation ● Demo video ● Making song arrangements