

FEBRUARY 10, 2023

MARBLES VR

MAXIME BUCK

TABLE OF CONTENT

1. Introduction	
1.1 Short project summary	2
1.2 How I got involved – LTS	2
1.3 What is VR?	2
1.3 Motivation	3
2. Project description	
2.1 Idea & Brainstorming	3
2.2 Hand-tracking / Controllers	4
2.3 Game Engine / Unity	4
2.4 Tools and Equipment	5
2.5 Planning	5
3. Project realization	
3.1 Oculus integration	6
3.2 Special hand tracking mechanics	7
3.3 3D modelling	8
3.7 Coding	
3.7.1 Grid & Pipes	9
3.7.2 Spawners	10
3.7.3 Menu & Canon	11
3.8 Testing & Problems	12
3.9 Solutions & Booster object	12
3.10 Optimization	13
4. Extras	
4.1 Some numbers	14
4.2 Game release / Conclusion	14
5. Appendix	
5.1 References	15
5.2 Word explanations	15
5.3 Image sources	18

1. Introduction

1.1 Short project summary

My project is the conceptualization and development of a VR puzzle game that combines hand-tracking and unique mechanics to challenge players to build a path with pipes, guiding a marble from start to finish. Simultaneously my goal is to provide a relaxing and immersive experience.

Before delving deeper into the details of my project, I will provide an overview of my background and motivations.

1.2 How I got involved – LTS

Discovering the national Jonk Fuerscher Contest Luxembourg was a defining moment in my journey as a student at LTS (Luxembourg Tech School)¹. As a third-year student, I have had the opportunity to discover various cutting-edge technologies and sciences, including game development, big data, AI and more. During one of my lessons, Edouard Olszewski, a representative from your federation, gave a captivating presentation on the competition instantly igniting my interest. The idea of showcasing my project to a wider audience and potentially winning one of the impressive prizes, such as a trip abroad, became my driving force. I eagerly pre-registered for the competition and began working towards this exciting opportunity.



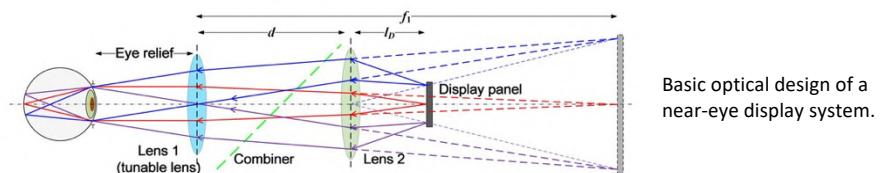
Me working on a project during a LTS session about space.

1.3 What is VR?

I have chosen to utilize virtual reality as the foundation for my project. In this section, I will dive into the exciting world of VR and explain why I am so captivated by its potential.

Virtual Reality, or VR, is a technology that employs 3D near eye displays to immerse the users in a digital environment. By wearing a VR headset, such as the Oculus Quest 2², users can experience a three-dimensional, computer-generated world that feels real. So as a tech enthusiast I have always been fascinated with the potential of VR. The ability to step into an entirely new world without physical constraints is simply mind-blowing. This allows VR to extend far beyond gaming, as it can be used for education, training, therapy and even art.

Overall, I believe VR has the power to change the way we interact with digital content, and I am impatient to see where this technology will go in the future.



1.3 Motivation

The moment I learned about game development during my second year at tech school was a turning point for me. In fact, our module on game development gave me the opportunity to learn about Unity³ and its capabilities, including visual scripting with BOLT⁴. The module culminated in a team competition where my team and I created the winning game, "DropZone". Additionally, the thrill of bringing my game to life and winning the competition sparked my interest in game development even further.



Group picture of my team during the gamedev competition.

So, after the competition, I was determined to take my game development skills to the next level. I started learning C#⁵, one of the coding languages used with Unity, which opened new possibilities for me in game creation.

As mentioned before, I am a VR enthusiast and I love to immerse myself in games like A Township Tale, Nock, or Population One⁶ that teleport you in another world. After some research, I found out that I can develop games with unity for Oculus Quest 2 which would combine my two biggest passions into one project.

2. Project description

2.1 Idee & Brainstorming

As I was returning from a vacation in France, I found myself with some free time in the car. Instead of simply passing the time, I took the opportunity to jot down some creative game ideas that came to mind. One specific idea captured my imagination - a puzzle game featuring a giant marble rolling through a complex system of pipes and obstacles, reminiscent of popular games like Cuboro or Gravitrax⁷.



The objective of the game could be either level-based, where the player must connect the start and end with a limited number of pipes, or it could be an open-ended experience, where players could build freely and unleash their creativity with an infinite number of objects. The beauty of this concept is that, unlike real-life puzzles, players would never run out of pieces, allowing them to construct grand and imaginative builds with ease.



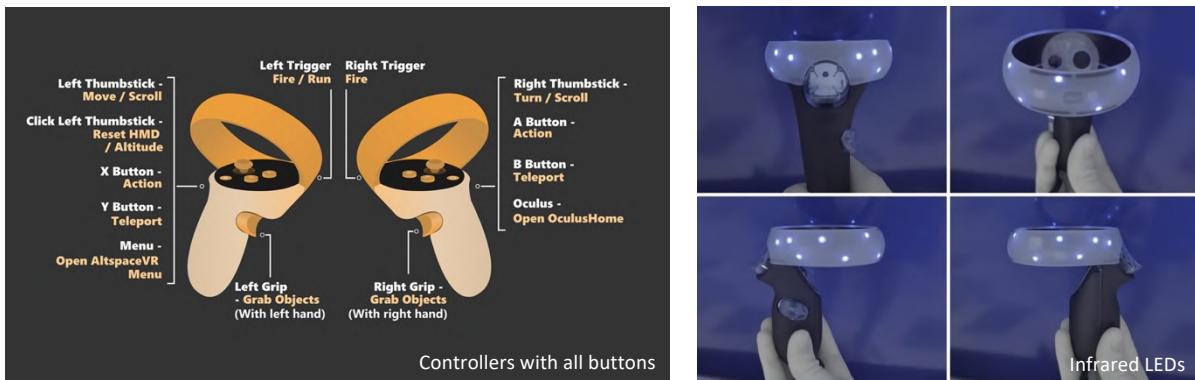
Refining the idea further, I realized that a 3D grid was necessary to ensure the seamless placement of objects and pipes. This grid would allow the player to align the objects in the right position, ensuring the smooth roll of the marble through the system without any overlapping or misalignment issues. Indeed, the lack of this 3D grid could potentially lead to frustration and cause players to give up on the game due to the high level of precision that would be required to place pipes.

Representation of a 3D grid

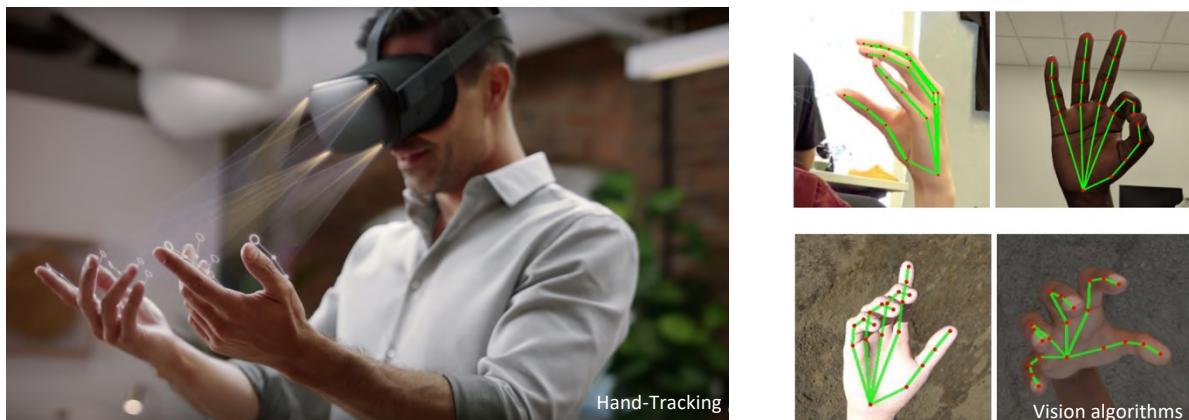
2.2 VR hand-tracking / controllers

How will the player interact with his environment? There are two distinct ways of engaging with VR games: controllers and hand tracking.

Controllers are physical devices held in the hands and used to send commands into the game such as moving, jumping or shooting. They work with different types of buttons and joysticks. Every controller also has a set of infrared LEDs⁸ located in the controller's rings allowing the headset cameras to track the position of your hands.



Hand tracking, on the other hand, allows players to use their hands to control the game directly without any physical device. This technology uses the inside-out cameras of the headset to track the position, orientation, and movements of the hands in real-time. It even perceives the configuration of your fingers using vision algorithms⁹.



After playing a little with all the possibilities, I decided to use hand-tracking for this game as this would permit unique mechanics using pose detection.

2.3 Game Engine – Unity

For my project, I needed to select a game engine¹⁰, which is a software platform specifically designed to help developers create video games. They provide a set of tools, libraries, and features that help to accelerate the process of building a game. But which one should I choose to make my VR game?



Unity

First, there is Unity, one of the most popular game engines in the world. It is a flexible and powerful engine that provides a wide range of tools and features, making it possible to create games for multiple platforms like

desktop, mobile, and most importantly VR and AR. Unity is also known for its large community of developers, therefore there are a lot of tutorials and documentation online that could help me on my journey.

Unreal Engine¹¹ is another popular game engine that is often compared to Unity. It is more powerful than Unity and is used to create high-end games. Indeed, Unreal Engine provides advanced features such as real-time rendering, advanced AI and physics simulation, making it the preferred choice for some developers aiming to make big AAA games¹².



However, this isn't really what I needed as those features are quite cost-effective in terms of computing power. Indeed, this VR headset's power is quite restricted. That's why, amongst other things mentioned before, I made the crucial decision to go with Unity over Unreal Engine. Besides I had already learned how to use it and was comfortable with its features and tools.

2.4 Tools and Equipment

	Tools	Name
1.	VR headset with hand-tracking capability	Oculus Quest 2
2.	Game engine with VR support	Unity
3.	Code editor	Visual Studio
4.	Computer	Mac
5.	A 3d modelling software	Shapr3d
6.	Planning application	Jira

2.5 Planning

Initially, minimal planning was necessary as the project was merely a leisure activity that I pursued in my spare time. As the project expanded, I required a way to keep track of tasks and future steps. I utilized an app called Jira for this purpose. I mainly used its Board feature, where different tasks could be categorized as "to do", "in progress", or "done".

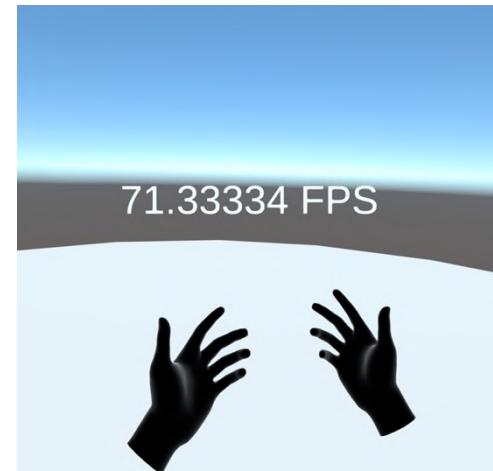
The screenshot shows the Jira Board interface with three columns: 'TO DO 6', 'IN PROGRESS 2', and 'DONE 5'. Each column has a header and a list of tasks with checkboxes and labels like 'MVR-25' through 'MVR-31'. The 'To Do' column includes tasks like 'Canon movement', 'Canon Garbbable calmp to up rotation', 'idle pipe flying in air emtying sound', 'pipes on spawner to know which one is which', 'Cant interact when playing', and 'let go if not in collider any more so cant grab handle and fly away'. The 'In Progress' column includes 'ball faster', 'Marble addforce', and a '+ Create' button. The 'Done' column includes 'Canon for UI', 'Menu', 'Grid', 'Spawners', '3D modelling', and another '+ Create' button. The bottom right corner of the interface says 'Jira - Board'.

3. Project realization

3.1 Oculus integration

To bring my VR game to life, I conducted extensive research and learned about the two ways to develop a VR game using Unity. The first is to use the XR interaction toolkit¹³, which is developed by Unity itself and has many features for almost all VR headsets. The other way is to use the Oculus Integration SDK¹⁴, which is provided by Meta. I decided to use the Oculus SDK as it allowed me to utilize the latest and more stable version of the hand-tracking technology.

To start the implementation of the VR technology, I had to ensure a smooth and immersive experience for the player. This involved addressing various key aspects, such as the player's view rotation, which had to match the rotation of their head movement. Additionally, I had to account for differences in player height and ensure that the game displays correctly for all players. Implementing hand tracking was a particularly challenging task, but once it was set up and working, it allowed me to display the black hands accurately on the player's virtual hands, as seen in the pictures. With these VR technology elements in place, I could then move on to the welding part of my project with confidence.



Afterwards, I added a sturdy platform to prevent the feeling of floating in an endless void and replaced the dull default sky with a more attractive one. Drawing from my prior experiences, I deemed it necessary to include an in-game FPS (frames per second)¹⁵ counter to aid in the development process before removing it in the final release.



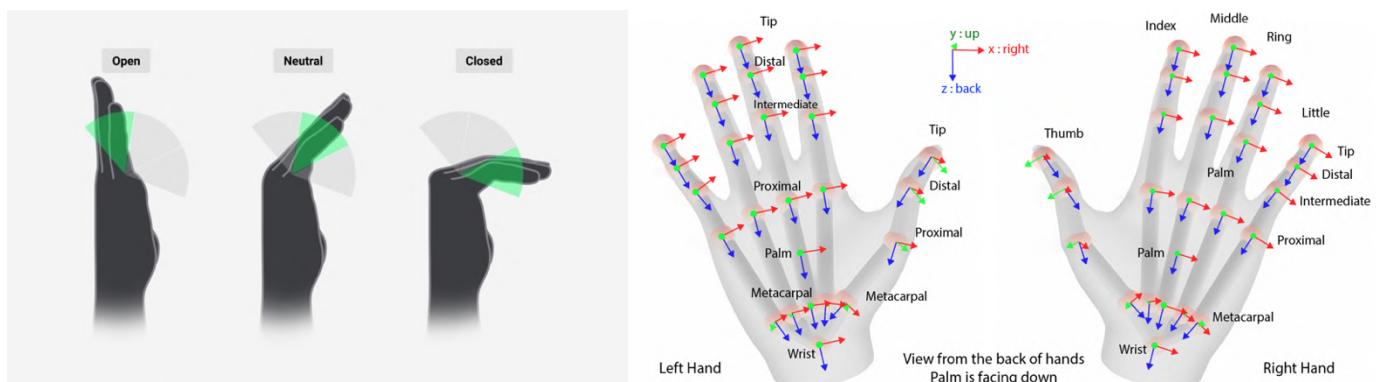
With the VR technology properly set up, the foundation of my project was firmly in place, and I was ready to build the core of my project and bring my vision to life.

3.2 Special hand tracking mechanics

For my project, I wanted to bring something new and innovative to the gaming industry by incorporating unique hand-tracking mechanics. With the hand tracking built into the Oculus SDK, I was able to detect the position of my fingers and determine if they were in a certain pose or performing a certain gesture. Thanks to this feature, I can detect poses like a bunny, a fist, or even a rock & roll pose. With that in mind, I thought about where I could use this in my project, and there were no limits to my creativity. Here's what I came up with:

A. Movement:

To allow the player to move around, I had to explore alternative options as traditional controllers rely on joysticks. I came up with a solution where the player would be accelerated in the direction he is pointing. As a result, the player has complete control over their movement and the ability to navigate the game world simply by turning their hand. But what was the implementation process behind this innovative idea?



1. I had to choose the position I wanted to recognize
2. Then I had to set the pose - I can specify, among other things, whether the fingers need to be closed, neutral or open and this for each joint.
3. Upon detection, the pose detector sends a message to a script attached to the player.
4. When the script receives the message, it accelerates the player in the direction your index finger is pointing

B. Marble Spawning:

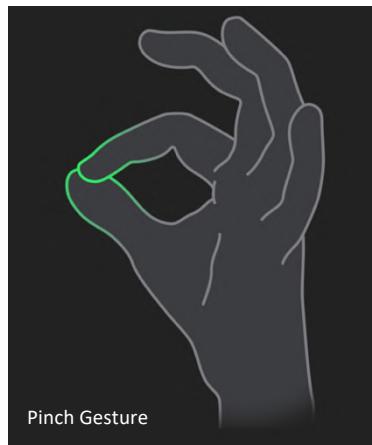
I was looking for a way to make the marble spawn when the player was satisfied with their parkour and wanted to test it out. I wanted the player to feel like they had supernatural abilities and could magically create things out of thin air. To achieve this, I decided that the player's hand pose would trigger the spawning of the marble. But how can you achieve that?



- This time the required pose is a flat hand with the palm facing up
- When this pose is detected, the following sequence occurs:
 - Some Particles appear above the hand.
 - If the pose is maintained, the particle animation continues for 2 seconds and a marble spawns and falls into the player's hand.
 - However, if the pose is interrupted, the particles disappear, and no marble is spawned.

C. Grabbing:

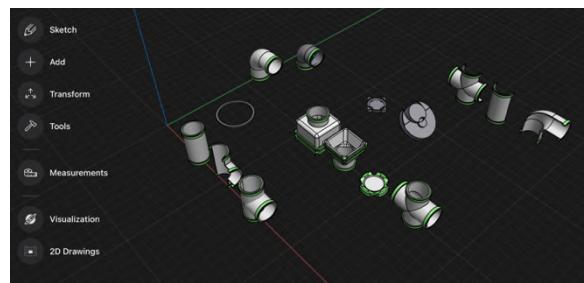
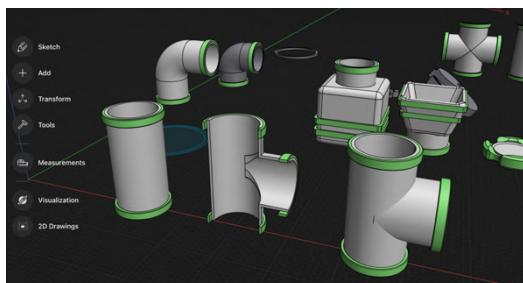
Integrating hand tracking into VR games is seamless with the Oculus SDK. The pre-built components provide an easy method for grabbing objects in your game. Players can use either the pinch gesture (pressing thumb and index finger together) or physically close their hand around the object to initiate the grabbing process. The code detects the intended grab and maintains the object attached to the player's hand until it is released.



3.3 3D modelling

To create the necessary 3D objects for my VR game, I used the application Sketch3D. Considering the limited computing capabilities of VR headsets, I opted for a low-poly approach and simplistic design. First, I had to improve my skills in Sketch3D with the help of tutorials and some sketches before embarking on creating the objects listed below:

- Straight pipe
- Curved pipe
- End pipe (serves as the goal)
- Start pipe (where you throw the marble in)
- Platform and a table



Sketch3D also provided me with the ability to experiment with different materials and colors for the objects. In the accompanying images you can see an example of using green for the pipe



and how it appears in the game with orange colors (from left to right: start pipe, end pipe, straight pipe, curved pipe):

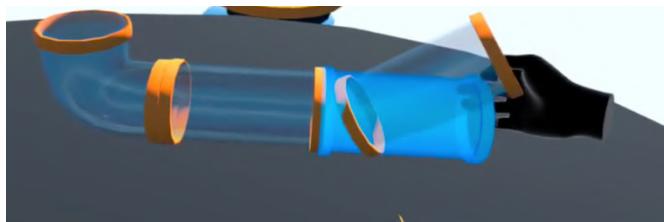


3.7 Coding

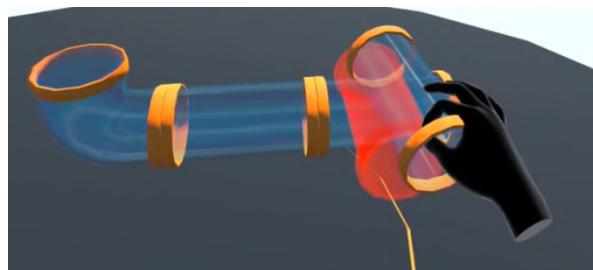
3.7.1 Grid & Pipes

I have already spoken about the grid I intended to implement. Here is a detailed explanation of how it operates:

- Above the table in the game, there is an area designated for pipe placement. When the player holds a pipe in this area, a hologram appears.
- The hologram accurately displays the position and rotation that the pipe would occupy once released. This gives the player a clear understanding of where the pipe will snap into place. Additionally, when the pipe is released, particles are spawned and a satisfying sound is played.



- As you can see in the following picture, to prevent players from placing pipes in occupied areas, the hologram turns red if it is held in a place that is already occupied. If the player releases the pipe in this case, an error sound is played and the pipe floats away.



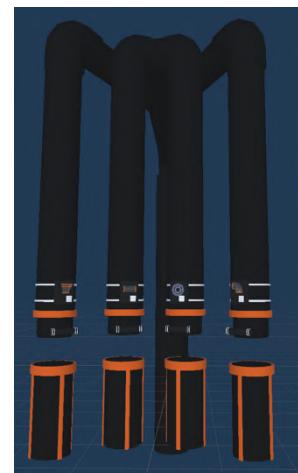
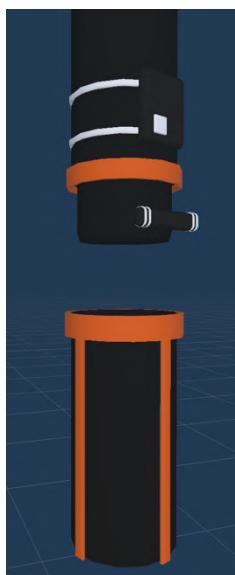
When the player is satisfied with his result, the pipe system can be tested by dropping a marble into the start pipe. If the parkour is connected in the right way, the marble rolls through the parkour until it drops in the end pipe.



3.7.2 Spawners

There has to be a way for the player to get more and more pipes, but I couldn't just place thousands of them on the floor. So, I came up with the idea of spawners that act as a source for new pipes.

I envisioned the spawner itself to resemble a hydraulic press found in factories. When you set it in action a pipe is spawned. Above each spawner, a sign indicates the type of pipe available and the number of pipes remaining. The counter is updated with each new pipe taken, and in sandbox mode, the count is set to unlimited. But what makes this possible?



- When the moving part is pressed below a certain threshold, some particles appear, a nice sound is heard, and a pipe is created.
- Then you must set it up again to make it work another time. This is to prevent hundreds of pipes from spawning when you hold the handle down without letting go of it.
- When you release the handle, it automatically jumps back up, this is done by kind of spring mechanism.

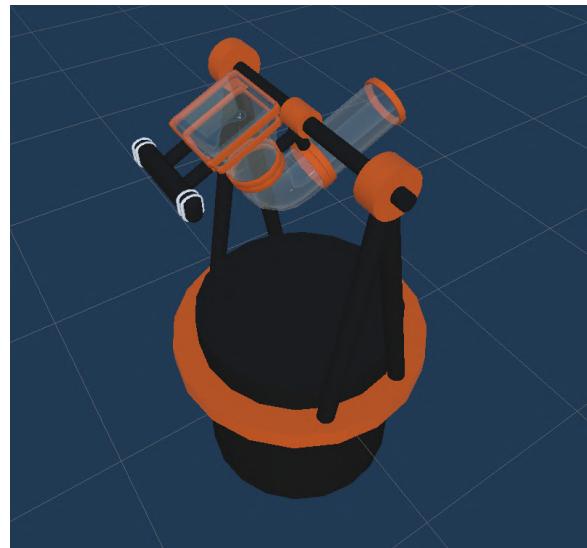


3.7.3 Menu & Canon

Compared to ordinary flat games, I had to be a little creative in designing the menu. I came up with the idea of a cannon that shoots marbles. Instead of a normal menu where you click here, it is the collision of the marble with the button that provides the interaction.

First I had to create this cannon. I did not need to model it, but I reused some pipes, the handle of the spawners and some cylinders.

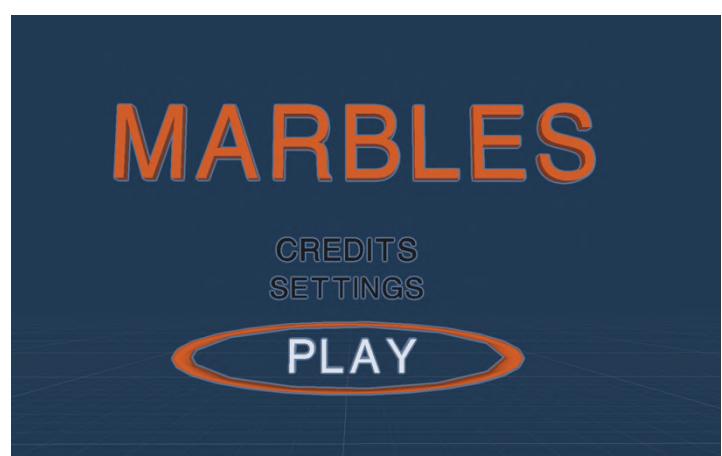
The advantage of this cannon is that you can swing it around completely freely.



When you hold the cannon in your hand, a red dot appears in the menu to indicate where you are aiming. If you move the mouse pointer over a button, it becomes larger.

To shoot the cannon, you must spawn a marble in your hand like in the normal game and drop it into the tube. Two different things happen in the cannon. The first is that the marble gets much bigger so that it looks like a cannonball, and you can see it clearly. The other is that a force is exerted in the direction the cannon is pointing.

As soon as a button is pressed, the whole menu drops down and the next one floats in. Of course, everything is accompanied by some juicy sound effects and particles to make the interaction with the player even more fun.



3.8 Testing & Problems

Once all these planned things were implemented into the game, my game went through some testing. While testing some issues were found:

- A. Sometimes while trying to grab an object, you would move, because the game though it had detected the pose. This was a big problem as it would get the player sick.
- B. As the Marble is moved through the parkour with gravity from the physics engine of Unity, the marble would sometimes get stuck. Indeed, when there is a long pipe without any drops, the marble will slow down and come to a. There also wasn't the possibility to do crazy stuff as the marble would just slow down.

3.9 Solutions & Booster object

Here are the solutions I came up with:

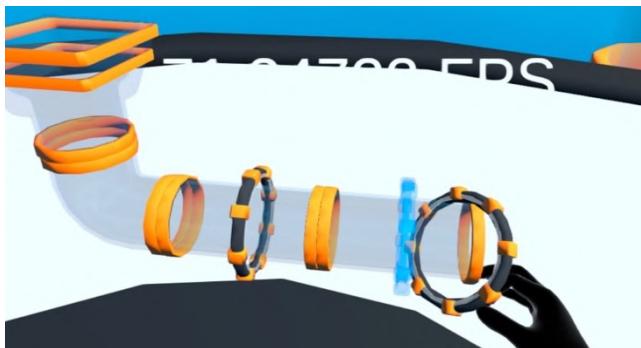
- A. To ensure that the player's hand movements do not interfere with the grabbing detection, I carefully selected a new pose for movement - straightening the hand. However, this brought up a new challenge as the marble spawning pose is the same just with your palm facing up. To resolve this issue, I devised a solution where each hand would deal with a specific task. One hand takes care of the movement and the other of the marble spawning, so they wouldn't interfere with each other.



Additionally, I included the option to customize the hand assignment in the game's settings menu. To accommodate different playing styles and preferences, I implemented the option to adjust the speed of movement. This way, players who may experience discomfort with a faster movement speed can adjust it to their liking, while more experienced players who prefer a faster pace can increase the speed to their desired level.

- B. The concept of adding a booster object came to my mind and after some sketching, I was thrilled by its design, that would perfectly fit with the game's aesthetic. Hereafter a picture of the booster in game after I 3d modeled it in Sharp3D.

The booster is designed to be placed around the pipe, providing an acceleration boost to any marble that rolls through it. This new addition opens up a wealth of possibilities for players, including the ability to construct a looping.



3.10 Optimization

To ensure the best gaming experience, optimizing the VR game is crucial. This includes reducing latency and maintaining a high frame rate to avoid motion sickness, dizziness and nausea.

During development, I focused on keeping the 3D models simple, with a small number of polygons, and limiting the use of textures. Shadows were used primarily for important objects, while they were disabled for others, such as the menu, as they were deemed unnecessary.

I also thoroughly reviewed my scripts to eliminate unnecessary code that was constantly running and drastically reducing the amount of FPS (frames per second)¹⁵.

4. Extras

4.1 Some numbers

- ± 200 hours spent on the game development
- ± 100 bugs fixed
- ± 1000 lines of code
- Zero cost except electricity (I had already all the other tools & equipment)
- 4 Unique features
- ± 10 hours spent on the report

4.2 Game release / Conclusion

Publishing a VR game is a significant moment in the development process and requires careful planning and execution to ensure its success. My plan for releasing the VR game involves the following steps.

First and foremost, I want to complete the development of the game and ensure that all features and content are complete and working as intended. This includes polishing the gameplay and fixing any remaining bugs or issues to give players the best experience possible. Once enough levels have been created and the overall experience has been optimized, I will be ready to proceed with the release.

In terms of distribution, I plan on releasing the game on the Oculus App Lab. This is a platform that allows developers to upload their games and make them available to the public for all oculus devices. After a review process, my game might even be available on the real oculus store.

To maximize the impact of the game's release, I think it is important to build a fan base before the game is released. A strong fan base can help spread the word about the game, raise its profile and attract more players. This can be achieved by posting videos of the game online, showcasing the game's features, gameplay and style, and by engaging with potential players to generate excitement and interest.

To sum up, my adventure with this project is not yet complete, but so far the development has been a success and the main concept was proven feasible.

5. Appendix

5.1 References

- Oculus SDK documentation
- YouTube tutorials: Brakeys, Valem, CodeMonkey, Justin P Barnett
- Unity forum
- Unity 3D documentation
- Book: *A theory of fun* by Raph Coster

5.2 Word explanations

1. LTS (Luxemburg Tech School)

<https://www.techschool.lu/>

Since 2016, the Luxembourg Tech School supports the national strategy by developing the future digital leaders. We implemented a unique project-based methodology that has proven, not only, successful, but also highly motivating and fun for you.

You work in teams on your own projects, learn new technologies and get personalized coaching! From Creative Coding to Game Development, passing by Financial Technologies, Artificial Intelligence or even Space, we provide a packed program.

2. Oculus Quest 2

https://en.wikipedia.org/wiki/Meta_Quest_2

Meta Quest 2 (initially sold as Oculus Quest 2) is a [virtual reality \(VR\) headset](#) developed by [Meta Platforms](#) (formerly Facebook, Inc.). It was unveiled on September 16, 2020 and released on October 13.

Meta website and store: <https://www.meta.com/>

3. Unity

[https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))

Unity is a [cross-platform game engine](#) developed by [Unity Technologies](#), first announced and released in June 2005 at [Apple Worldwide Developers Conference](#) as a [Mac OS X](#) game engine. The engine has since been gradually extended to support a variety of [desktop](#), [mobile](#), [console](#) and [virtual reality](#) platforms. It is particularly popular for [iOS](#) and [Android](#) mobile game development, is considered easy to use for beginner developers, and is popular for [indie game](#) development.^[6]

4. BOLT / visual scripting

<https://assetstore.unity.com/packages/tools/visual-scripting/bolt-163802>

Bolt brings complete visual scripting to Unity, empowering artists, designers and programmers to create gameplay mechanics and interactive systems without writing a single line of code.

5 C#

[https://en.wikipedia.org/wiki/C_Sharp_\(programming_language\)](https://en.wikipedia.org/wiki/C_Sharp_(programming_language))

C# (pronounced C sharp)^[15] is a general-purpose [high-level programming language](#) supporting multiple [paradigms](#). C# encompasses static typing, [strong typing](#), [lexically scoped](#), [imperative](#), [declarative](#), [functional](#), [generic](#), [object-oriented](#) ([class-based](#)), and [component-oriented](#) programming disciplines.^[16]

The C# programming language was designed by [Anders Hejlsberg](#) from [Microsoft](#) in 2000 and was later approved as an [international standard](#) by [Ecma](#) (ECMA-334) in 2002 and [ISO/IEC](#) (ISO/IEC 23270) in 2003.

6. A Township Tale, Nock, Population One

A Township tale:

<https://townshiptale.com/>

Nock:

https://www.oculus.com/experiences/quest/5157404804284116/?locale=de_DE

Population One:

<http://www.populationonevr.com/>

7. Cuboro / Gravitrax

Cuboro:

<https://cuboro.ch/en/>

Gravitrax:

https://www.ravensburger.fr/produits/gravitrax/gravitrax-sets-d-extension/index.html?_ga=tsid:96596&gclid=CjwKCAiArY2fBhB9EiwAWqHK6p2bhKX2RDGljVG6PtOzkxDpvXgGR8RYj9Ngqbp3jsZy7_ZF3mtghxoCmTwQAvD_BwE

8. Infrared LEDs

<https://de.wikipedia.org/wiki/Infrarot-LED>

An infrared LED (short IR-LED or IRED) is a special light-emitting diode (LED, therefore also infrared diode), which emits light in the [near-infrared range](#) with a [wavelength](#) of 700 nm to 1000 nm. This range of the light spectrum is not visible to the human [eye](#), but can be measured with [radiation detectors](#) made of [pyroelectric](#) materials or [semiconductors](#), such as [photodiodes](#) or [phototransistors](#). Digital cameras are sensitive to both visible and infrared light unless an infrared filter is installed. With their help, for example, the functionality of [infrared remote controls](#) can be easily checked.

9. Vision algorithms:

<https://www.run.ai/guides/deep-learning-for-computer-vision#:~:text=Computer%20vision%20algorithms%20analyze%20certain,commonly%20used%20for%20computer%20vision>

Computer vision algorithms analyze certain criteria in images and videos, and then apply interpretations to predictive or decision-making tasks.

<https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e>

10. game engine:

https://en.wikipedia.org/wiki/Game_engine

A game engine is a [software framework](#) primarily designed for the development of [video games](#) and generally includes relevant [libraries](#) and support programs.^[1] The "engine" terminology is similar to the term "[software engine](#)" used in the [software industry](#).

11. Unreal Engine:

https://en.wikipedia.org/wiki/Unreal_Engine

Unreal Engine (UE) is a [3D computer graphics game engine](#) developed by [Epic Games](#), first showcased in the 1998 [first-person shooter](#) game [Unreal](#). Initially developed for [PC](#) first-person shooters, it has since been used in a variety of genres of games and has seen adoption by other industries, most notably the film and television industry.

Unreal engine website: <https://www.unrealengine.com/en-US>

12. AAA games:

[https://en.wikipedia.org/wiki/AAA_\(video_game_industry\)](https://en.wikipedia.org/wiki/AAA_(video_game_industry))

In the [video game industry](#), AAA (pronounced and sometimes written triple-A) is an informal classification used to categorise games produced and distributed by a [mid-sized](#) or [major publisher](#), which typically have higher development and marketing budgets than other tiers of games.^[1]

13. XR interaction toolkit:

<https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.2/manual/index.html> :

The XR Interaction Toolkit package is a high-level, component-based, interaction system for creating VR and AR experiences. It provides a framework that makes 3D and UI interactions available from Unity input events.

14. Oculus integration:

<https://developer.oculus.com/documentation/unity/unity-isdk-interaction-sdk-overview/>

SDK provides a library of components for adding controllers and hand interactions to your experiences. This includes interaction models such as ray, poke, and grab, which incorporate best practices and heuristics for user interactions on Meta Quest devices.

15. Frame rate

https://en.wikipedia.org/wiki/Frame_rate

Frame rate (expressed in frames per second or FPS) is the [frequency](#) (rate) at which consecutive [images](#) ([frames](#)) are captured or displayed. The term applies equally to [film](#) and [video cameras](#), [computer graphics](#), and [motion capture](#) systems.

5.3 Image sources

- Page 2: Near-Eye Display System: https://www.researchgate.net/figure/Basic-optical-design-of-our-near-eye-display-system_fig3_335144486
- Page 3: Gravitrax: <https://www.amazon.com/Gravitrax-Vertical-Starter-Pack-26832-GraviTrax/dp/B084DGQKXD>
3D Grid: <https://www.vectorstock.com/royalty-free-vector/cube-3d-mesh-wireframe-web-and-data-connection-vector-17015141>
- Page 4: Controllers: <https://learn.microsoft.com/en-us/windows/mixed-reality/alspace-vr/getting-started/oculus-controls>
Infrared LEDs: https://www.reddit.com/r/OculusQuest/comments/ca28kn/15_tracking_dots_per_touch_controller_seen/
Hand-Tracking: <https://vrx.vr-expert.com/de/die-bedeutung-von-hand-tracking-fuer-vr-ar-unternehmen/>
Vision algorithms: <https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html>
- Page 5: Unity: https://de.wikipedia.org/wiki/Unity_%28Spiel-Engine%29
Unreal Engine: https://commons.wikimedia.org/wiki/File:Unreal_Engine_Logo.svg
- Page 7: Shape Recognizer: <https://developer.oculus.com/documentation/unity/unity-isdk-hand-pose-detection/>
Hands: <https://docs.godotengine.org/en/stable/tutorials/vr/openxr/handtracking.html>
- Page 8: Pinch Gesture: <https://developer.oculus.com/documentation/unity/unity-isdk-hand-pose-detection/>