MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Project Tsunami: Alternative Teleportation in VR

Bachelor's Thesis

TIMOTEJ DŽUGAS

Brno, Fall 2023

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Project Tsunami: Alternative Teleportation in VR

Bachelor's Thesis

TIMOTEJ DŽUGAS

Advisor: Ing. Matěj Lang

Department of Visual Informatics

Brno, Fall 2023

MUNI
FI

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Timotej Džugas

Advisor: Ing. Matěj Lang

Acknowledgements

I am immensely grateful to my advisor, Ing. Matěj Lang, for his invaluable guidance, advice, and feedback throughout this thesis. Additionally, I am thankful for the assistance provided by my consultant, RNDr. Vít Rusňák, Ph.D., in designing the user study. I am also grateful to the HCI Laboratory on FI MU for lending me a VR headset. Lastly, I would like to extend my appreciation to all the participants who took part in the user study.

Abstract

Traditional VR locomotion techniques such as Point and Teleport or joystick are not very efficient for navigating in a big open VR environment. To tackle this problem, this thesis presents a new virtual reality locomotion technique called Tsunami, which curves the surrounding environment towards the user in order to reveal previously inaccessible locations. It is a modification of the Point and Teleport Technique, which was chosen because of its simple controls and low chance of introducing motion sickness. It was simultaneously chosen as a baseline method for a user study that compared it to the two variations of Project Tsunami. The study with 12 participants revealed that the Tsunami techniques use significantly fewer teleports to cover the same distance. The time efficiency and accuracy results were inconclusive and require an extra future analysis. As it is a pilot study, comparisons between the Tsunami variations were also made and showed that the Tsunami without delay is a more viable option in terms of both performance and preference for future use.

Keywords

VR, locomotion, unity, user study, pilot study

Contents

Introduction **1**

1 Related Work **2**
1.1 VR Locomotion 2
1.2 Locomotion Techniques 2
1.2.1 Point and Teleport 3
1.3 Teleport Techniques Comparison 4

2 Design **7**
2.1 Initial Idea 7
2.2 Curving the Environment 8
2.2.1 Curving the Plane 9
2.2.2 Curving the Objects Above the Plane 10
2.3 Collisions with the Curved World 11

3 Technology Used **14**
3.1 Oculus Quest 2 14
3.2 Unity 15
3.2.1 Classes and Components 15
3.3 XR Interaction Toolkit 16
3.4 OpenXR 16

4 Implementation **17**
4.1 Surface Shaders 17
4.2 Curving the Environment 18
4.3 Collisions with the Curved World 20

5 User Study **27**
5.1 Disclaimer 27
5.2 Participants 27
5.3 Tested Locomotion Methods 28
5.4 Hypotheses 28
5.5 Apparatus 29
5.5.1 The Testing Application 29
5.5.2 Questionnaires 32

5.5.3 Learning Effect 33
5.6 Procedure 33
5.7 Data Processing 34

6 Results **36**
6.1 Accuracy Test 36
6.2 Speed Test 38
6.2.1 Teleport Count 39
6.3 User Feedback 42

7 Discussion **45**
7.1 Accuracy 45
7.2 Speed 45
7.3 Teleport Count 46
7.4 Motion sickness 47
7.5 Suggestions 47

8 Limitations **48**

9 Conclusion **49**

Bibliography **51**

A An appendix **54**
A.1 Appended Files 54
A.2 Consent form 54
A.3 Initial Questionnaire 54
A.4 Middle Questionnaire 55
A.5 Final Questionnaire 56

List of Figures

1 VR locomotion typology presented by Boletsis et al. [2]. . 3
2 Point and Teleport presented by Boletsis et al. [2]. 4
3 Unity's left-handed coordinate system (source) 8
4 Transformation f_1 9
5 Abstraction of transformation f_2 on plane bent only in z direction 10
6 Pointer ray collisions with standard parabola 12
7 Pointer ray after "flattening" 12
8 Abstraction of construction of a curved pointer ray 13
9 Oculus Quest 2 controller and buttons used in this thesis 14
10 A plane transformed by the `TsunamiPlane.shader` 19
11 Screenshot of `TsunamiComplex.shader` from Unity editor 20
12 Screenshot of a Training ground 30
13 In-game screenshots of an accuracy test with a long-range target (left: baseline, right: Tsunami) 31
14 In-game screenshot of a speed test (left: baseline, right: Tsunami) 32
15 A participant in the middle of the user study 35
16 Box plot for each of the tested accuracy ranges 36
17 Speed Test times box plots 38
18 Number of teleports for each path in Speed Test 40
19 Visualised paths for each method in one run 41
20 Evaluation of ranking questions from the final questionnaire 42
21 Preference of the methods at short/mid/long range 43
22 Evaluation of Likert questions from the middle questionnaire 44

Introduction

Virtual reality (VR) is becoming increasingly popular in society as a result of continually evolving and more affordable technologies that make VR technology accessible to the general public. It has revolutionized the way people experience virtual worlds, offering immersive environments from the comfort of their homes.

Among the many challenges VR developers have to face, locomotion remains a crucial aspect that greatly affects the user experience. The main limiting factor is the physical space the user has available. While traditional Point and Teleport (P&T) [1] has been the primary locomotion technique used in VR applications, offering a solution to motion sickness and being relatively intuitive and simple to use; this thesis proposes a new alternative that will enhance this technique.

The main idea of Project Tsunami is to make the classic P&T more efficient in open VR environments such as cities or nature by curving the surroundings towards the user and providing access to previously unreachable places, which should result in cutting the time the user has to spend navigating through dense but open environments. This claim will be validated in a user study that compares Project Tsunami with the baseline P&T in terms of both time efficiency and accuracy.

The thesis contains a review of research papers on related work that present different VR locomotion techniques and their typology and comparisons. The following chapters cover the design and implementation of Project Tsunami. Additionally, the thesis covers the design of the user study, its results, and the ensuing discussion.

1 Related Work

This section discusses the concept of VR locomotion, focusing on the impact on user interaction and experience with motion sickness. It also highlights a literature review on locomotion method studies introducing a new VR locomotion classification typology. The section also discusses a study that compared the most common locomotion methods with insights into their benefits and challenges regarding user experience, immersion, and efficiency. Since this thesis aims to improve the P&T technique, the reader will be acquainted with this method more specifically.

1.1 VR Locomotion

In terms of virtual reality, the virtual locomotion technique refers to the techniques used to move and navigate in a virtual environment. It is a crucial component that heavily affects how user experiences and interacts with the environment. Therefore, many methods have been developed to address the problem of translating physical movements into virtual reality in a limited available physical space without causing motion sickness.

1.2 Locomotion Techniques

Boletsis et al. [2] conducted a review of the literature on studies published between 2014 and 2017 that assessed VR locomotion techniques. Thirty-six relevant articles, which evaluated at least one locomotion technique and conducted a user study, were selected, resulting in 73 instances of 11 unique VR locomotion techniques. The paper provides a summary of these techniques and makes recommendations for future work. It also suggests a new typology for VR locomotion techniques to aid in the classification of these methods. The two main categories that depended on interaction are physical, such as walking-in-place, redirected walking, arm-swinging, or real-walking, and artificial techniques: joystick, chair-based, head-directed, and teleportation. Each has its own sub-categories, as can be seen in Figure 1.

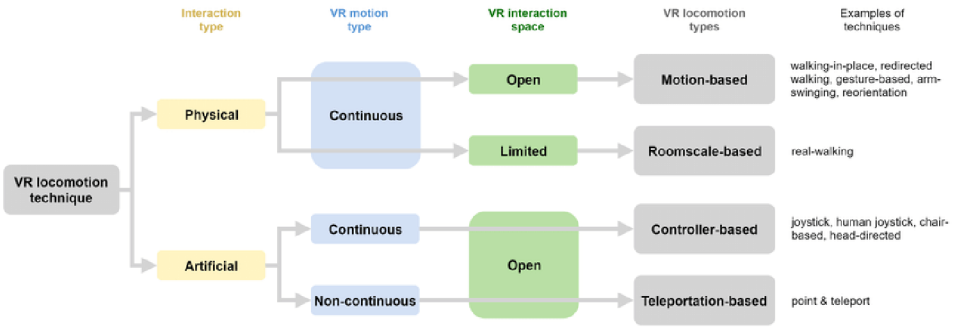

Figure 1: VR locomotion typology presented by Boletsis et al. [2].

1.2.1 Point and Teleport

In virtual reality, the Point and Teleport method is a locomotion method that allows users to navigate within a virtual environment by pointing to a specific location and then instantly teleporting to a new position. A straight or parabolic line extends from the controller while the user presses the teleportation button. The user can teleport to a new location indicated by the ray's collision position with the virtual world. Generally, the range and areas for teleportation are limited based on the type of application.

This technique and term *Point and Teleport* was introduced by Bozgeyikli et al. [1] The main advantage of P&T is that it introduces little to no motion sickness but at the expense of breaking the immersion. They conducted two experiments comparing this method to walking-in-place and joystick and a second experiment that added a direction component, in which the user could specify the direction they would be facing after the teleportation.

The first experiment's objective was to compare these techniques purely in terms of VR locomotion, with the only goal being navigation. Therefore, the environment was very simple, and no interaction tasks were introduced, which resulted in none of the techniques causing motion sickness. In terms of required effort and difficulty operating, both joystick and P&T over-performed walking-in-place. Moreover, tests were conducted to compare travel times to reach a destination, showing P&T to be the fastest in an environment without obstacles. However, when obstacles were introduced, the joystick performed

better than P&T. The authors discussed this finding and interpreted it as it was most probably caused by a two-second period the user had to be aiming at a new position to teleport. They suggest the use of instant teleportation instead.

The second experiment, which added the possibility of specifying direction after teleportation, proved to add no extra usability and even made the experience worse.

The Point And Teleport is depicted in the Figure 2

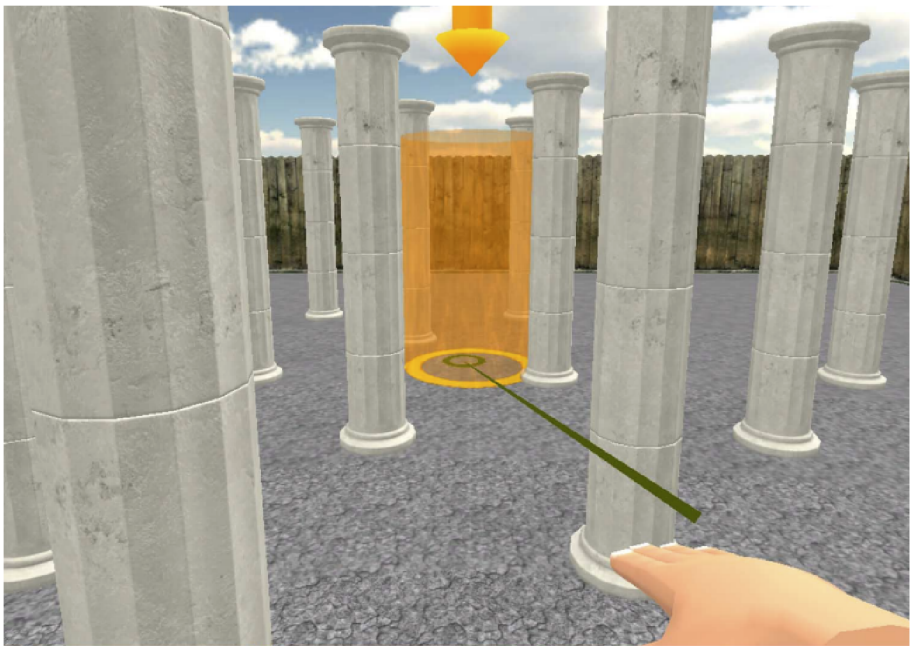

Figure 2: Point and Teleport presented by Boletsis et al. [2].

1.3 Teleport Techniques Comparison

Boletsis et al. [3] also conducted a study where they compared three of the aforementioned locomotion methods: walking-in-place, joystick, and teleportation. The experiment consisted of navigating through a virtual city in order to find a sequence of checkpoints. The findings demonstrate that whereas physical techniques, such as walking-in-place, offer a greater degree of immersion than artificial methods, they also frequently result in a considerable degree of psychophysical

discomfort. Moreover, the artificial approaches are considered easier to use because people are more accustomed to using controllers than moving physically to control their movement in a virtual environment. Teleportation is particularly useful since it allows quick positional changes over longer distances. However, these movements often interfere with the users' sense of immersion in a virtual environment due to their noncontinuous nature of motion. The results also show how similar the teleportation and joystick methods are in the majority of the factors assessed, including competence, challenge, and fatigue; nevertheless, because of its advantages in more effective navigation, teleportation has been demonstrated to be more effective than the joystick approach, which generates a more significant amount of motion sickness.

In 2021, a paper by Prithul et al. [4] was written about various teleportation methods. The paper provides a systematic review of improved teleportation techniques and presents results from user studies comparing them to different locomotion techniques. The results from user studies confirm the findings by Boletsis [3] that teleport causes a negligible amount of motion sickness. In most of the studies comparing locomotion techniques, a navigation or search task has been set to be the main task.

The only study that measured a mild amount of motion sickness was done by Líndal et al. [5]. They compared a teleport with a tracking system that moved users through the VE on a pre-defined path. The teleportation system was designed to determine the ray's direction by the forward direction of a head-mounted display (HMD). Only a marker indicating the new position on the ground was displayed, which is an unusual implementation compared to the standard teleportation system using controllers that are used in other studies. However, the method still showed fewer sickness symptoms than the tracking technique.

Prithul also discusses a few drawbacks of teleportation as a locomotion technique. The first issue is the non-continuous motion of the teleportation. This creates difficulties in interactions between players in multi-user environments such as first-person shooter games, where avatars would instantly appear and disappear, which would cause an undesirable experience. The second disadvantage is the previously

mentioned breaking of the immersion, confirmed by the results of the included studies.

The study found that some experiments compared the performance of the enhanced teleportation method with the regular P&T, but no significant difference was observed. However, Prithul points out that this does not necessarily mean that the two methods are equivalent, given the small number of participants in the user studies.

None of the improved methods surveyed by Prithul nor studies mentioned outside this thesis resembles the proposed alteration by Project Tsunami, which makes it a new and unique addition to the virtual locomotion technique field.

2 Design

In this chapter, the origin of Project Tsunami is described, along with the problems that were faced during the initial proof of concept. The theory behind curving the virtual environment is also discussed, and two important transformations that are used throughout this thesis are introduced. The reader will be provided with a clear understanding of the curving process, and the challenges that were overcome during the development of Project Tsunami will be discussed.

2.1 Initial Idea

Ing. Matěj Lang introduced the initial idea of Project Tsunami within the PV160: Human-Computer Interaction Laboratory. The project aimed to implement an alternative teleportation system in VR that curves the landscape upwards with every start of the teleportation sequence to select a new destination. The implementation was successful. However, it was missing functionality like collisions outside the curved plane, which allowed users to teleport through objects such as buildings. Moreover, the collisions worked only with flat ground and more complex terrains like hills were not supported.

The idea was to treat the virtual ground as a parabola and the teleportation ray as a line in three-dimensional space and find their intersection, which would represent a new position. This approach proved to work but at the expense of ignoring objects above the curved plane. The project was then extended as a bachelor thesis to rework the system and add the missing functionality.

Since the implementation of Project Tsunami is created in Unity, the thesis will be using its left-handed coordinate system where the positive x-axis points to the right, the positive y-axis points up, and the positive z-axis points forward, shown in Figure 3. Similarly, measurement units in meters will be used, where one meter equals the one measurement unit in the Unity game engine. Since the environment is not on a realistic scale, a distance considered as short is equal to 30 meters in this thesis.

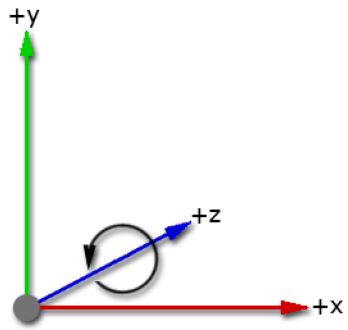

Figure 3: Unity's left-handed coordinate system (source)

2.2 Curving the Environment

This section discusses the theory behind the two object transformations that create a curvature effect in a virtual environment. These transformations are implemented using computer graphics shaders, which require calculation for each point of the object. Therefore, when referring to the geometry of an object, the term *vertex* is used in the thesis instead of *point*. Notably, shader knowledge is not necessary for this section and will be covered in the Implementation chapter 4.

A vertex is a term used in computer graphics and is defined as a point in 2D or 3D space containing multiple information pieces, including its position coordinates, colour, and texture. These vertices serve as fundamental units to create complex 3D objects by connecting them to form triangles that form complete shapes.

In this representation, a simple cube would be constructed using 8 vertices. As a result, a transformation would need to be applied 8 times.

2.2.1 Curving the Plane

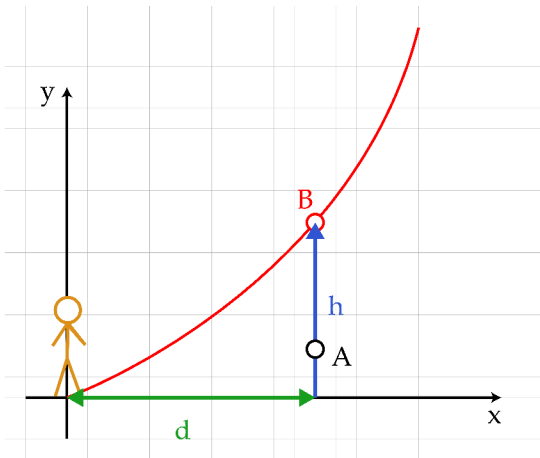

Figure 4: Transformation f_1

The first transformation f_1 maps vertex A to the appropriate position B on the parabola, as seen in Figure 4. The idea is that the height h by which the position of a vertex A is to be raised is calculated by raising the distance d from the user (curvature origin) to the power of k and multiplying it by a *curve coefficient*, forming a formula:

$$y = d^k * curvecoefficient \tag{2.1}$$

The curvature coefficient simultaneously controls the curvature amount of the world, which is utilised during the teleportation sequence.

2.2.2 Curving the Objects Above the Plane

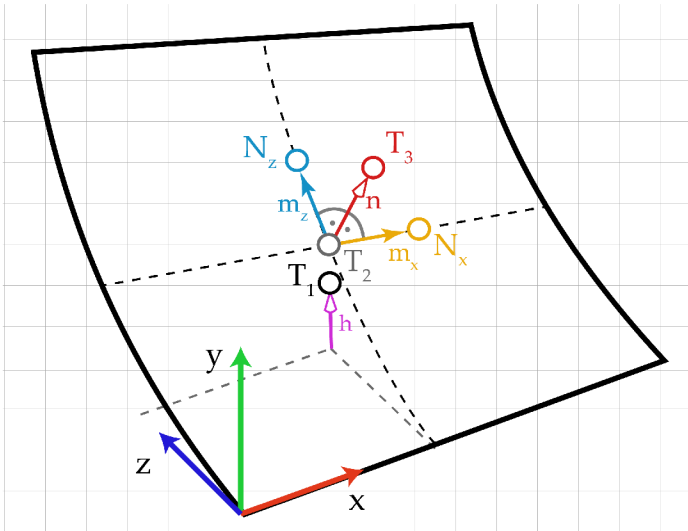

Figure 5: Abstraction of transformation f_2 on plane bent only in z direction

Vertices above the ground are transformed using transformation f_2. It is depicted in the Figure 5; because of simplicity, the depicted plane is bent only in z direction.

The first step is to set the y-coordinate of T_1 to the appropriate position on the parabola. That is achieved by initially applying the $f1$ on T_1, resulting in point T_2. To move the vertex above the parabola, a normal vector needs to be calculated perpendicular to the parabola from the position of T_2.

This is done by creating two auxiliary points N_x and N_z, which are calculated by adding a tiny value to the x and z coordinates of vertex T_1, respectively, and applying the $F1$ transformation. This creates a set of three points: T_2, N_x, N_z positioned on the surface of the parabola. Two vectors, m_x, m_z are then calculated from the point T_2 to its neighbours N_x, N_z.

$$m_{x,z} = N_{x,z} - T_2 \tag{2.2}$$

The normal vector n is calculated as a cross-product of m_z and m_x.

$$n = cross(m_z, m_x) \tag{2.3}$$

This vector is normalized and multiplied by an initial height h of the vertex T_1. Point T_2 is shifted along this vector, resulting in the final "curved" vertex T_3.

$$T_3 = T_2 + n * h \tag{2.4}$$

This procedure is calculated for each vertex of the object above the ground, resulting in it being positioned above the parabola, transforming the objects into the *curved world*.

2.3 Collisions with the Curved World

Unity does not permit manipulation of an object's collider mesh during run-time, so the effect is purely visual. This implies that the *curved world* seen by the user does not correspond to the object's actual position in the virtual environment. Since the positional data of the *curved world* are absent, it is not possible to find the collision in this world without additional calculations in the background. This finding was crucial and needed to be worked around in order to implement a fully functional Project Tsunami teleport.

Multiple approaches were tried, such as finding a way to modify collision mesh or a mapping function that would map the curved points to their original position, but none were successful.

The advisor of this thesis proposed an idea to calculate angles under which the invisible ray pointing forward from the controller (*pointer ray*) collides with the parabola to create a naive inverse approximation of the curving function by "flattening" the parabola and curving the pointer ray accordingly.

In other words, a standard parabola is utilised in the background to calculate auxiliary angles of the colliding pointer ray, ignoring the curved virtual environment. Then, the curved version of the pointer ray is utilised in the original, not-curved world (*flat world*) to find a collision with the object's collider mesh. The idea is depicted in two steps in Figures 6, 7 and explained below.

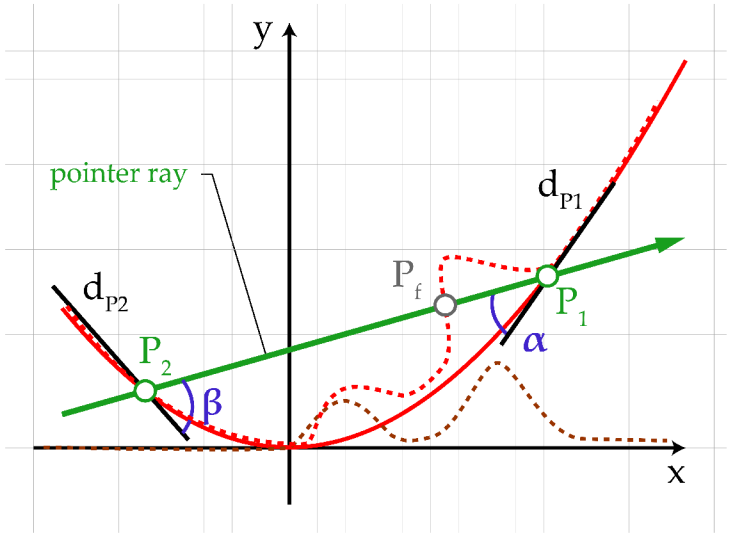

Figure 6: Pointer ray collisions with standard parabola

In Figure 6, the brown dashed line represents a flat world, while the red is the curved world. The green arrow represents the pointer ray.

The first step is to find the values of the α and β. They are calculated as angles between derivations d_{P1},d_{P2} in P_1,P_2 and the pointer ray respectively.

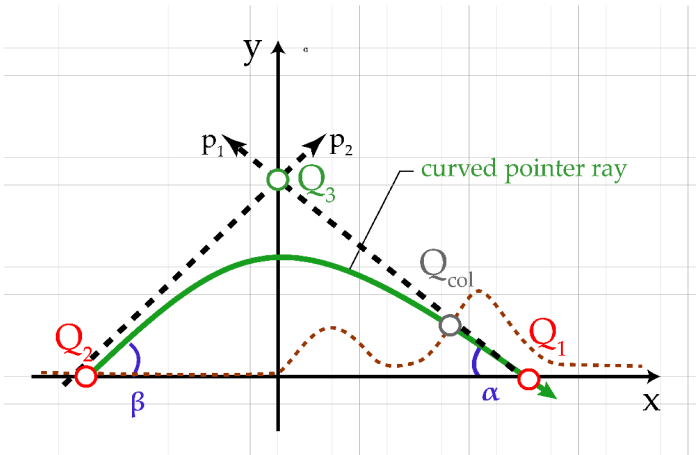

Figure 7: Pointer ray after "flattening"

Figure 7 represents the second step and depicts the flattened parabola as the x-axis and *pointer ray* after the "flattening". By keeping the values of α,β, two lines p_1,p_2 facing away from the plane are projected under these angles from the points Q_1, Q_2 and their intersection Q_3 is calculated. The curve is then constructed by interpolation between Q_1, Q_3, and Q_2, which creates the approximated curved pointer ray acting like a naive inverse function. This process creates a quadratic Beziér curve, and one with 5 steps of interpolation is approximately depicted in Figure 8.

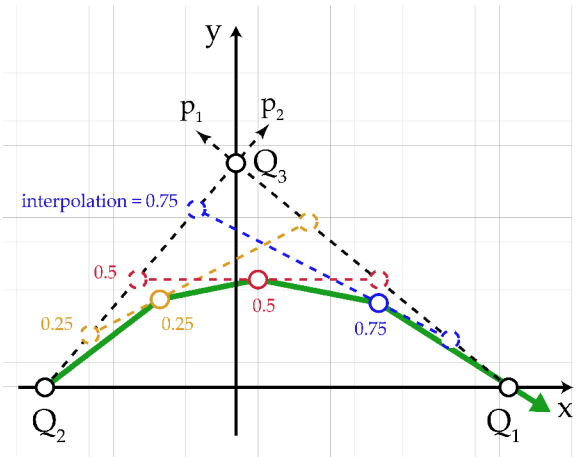

Figure 8: Abstraction of construction of a curved pointer ray

The coordinates of the collision with the terrain mesh and final position Q_{col} in the flat world can be easily calculated using Unity's integrated components, which will be covered in the Implementation chapter 4.

After calculating the final position Q_{col}, the *curve origin* and user's position are set to these coordinates in order to calculate the curvature of the environment from this position; therefore, the transformation f_1 will measure the distance of the vertices from these new coordinates.

This process maps the collision position of the pointer ray and the curved world P_f depicted in the Figure 6 to the flat world, resulting in the final position Q_{col}.

3 Technology Used

In this chapter, the technologies used to implement Project Tsunami, as well as the hardware used in user study, are described. It also covers the packages and components used within the Unity Engine to achieve the desired functionality.

3.1 Oculus Quest 2

This implementation uses an Oculus Quest 2 [1] virtual reality headset with Quest 2 Touch hand tracking controllers shown in the Figure 9.

The HMD's position and rotation are tracked in space using an accelerometer, gyroscope, and cameras integrated into the headset. Similarly, the controllers' position and rotation are tracked using similar techniques, providing users with virtual "hands" corresponding to real positions in the virtual space.

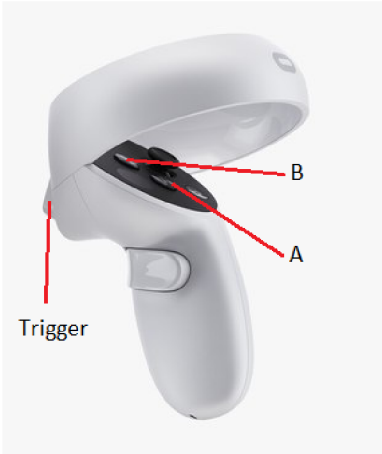

Figure 9: Oculus Quest 2 controller and buttons used in this thesis

1. https://www.meta.com/quest/products/quest-2/

3.2 Unity

Unity [2] is a game engine that offers a versatile platform for developing either desktop or mobile applications. It operates on the principles of a component-based architecture, where *Game Objects* [6] are the main building blocks constructed by assembling *Components* [7] that define their behaviour, appearance, and functionality. Game Objects are placed in *Scenes* [8], which incorporate a hierarchy system, creating a node system for parent-child relations between them. The scenes can be interpreted as levels, where each level in a game usually has its own scene.

3.2.1 Classes and Components

Unity provides a lot of integrated functions and components that can be easily used and customized. For example, each Game Object has a Transform [9] component, which holds information about the position, rotation and scale. It can be accessed through a script component in order to move, rotate or resize the Game Object.

The C# Scripts [10] are utilised to add custom functionality, with access to other components attached to the same Game Object, such as a Camera, Mesh Renderer or Collider. The Collider [11] component creates a collider mesh from an object's mesh, which is used for collision detection with other components, providing them with necessary data.

Unity also provides numerous classes, including a Physics class [12], which implements a framework for ray-casting and ray collisions.

A Ray [13] is a simple class with an origin and direction. It can be cast using a RayCast [14] method with a specified length, and in case of a collision with a Collider component, it returns a RaycastHit [15] object. It holds information about the collision, such as position, normal of the hit surface, or even the whole collider component of the Game Object, the ray hit, which can be easily accessed.

2. https://unity.com/

3.3 XR Interaction Toolkit

The XR Interaction Toolkit [16] package is a high-level package for Unity that implements an interaction system for creating VR and AR applications. It incorporates a framework that works with Unity's Interaction system and provides input data from the Oculus VR controllers. This package also comes with integrated components for locomotion, which contains f.e. components for P&T logic and visuals used in this thesis.

3.4 OpenXR

Another package used in this thesis is a Unity OpenXR plugin [17]. It is a standard that Khronos [3] created with the objective of making development easier by enabling users to target a variety of AR/VR devices.

3. https://www.khronos.org/

4 Implementation

In this chapter, the implementation of Tsunami locomotion will be covered. Firstly, the curvature of the environment using transformations f_1 and f_2 covered in the section 2.2 will be shown in the form of pseudo-code within the Surface shaders. Then, the implementation of the idea behind finding the corresponding collision in the flat world will be described using the samples of pseudo-code from Project Tsunami's implementation.

4.1 Surface Shaders

As previously mentioned, shaders only visually transform the objects, so all information about mesh vertices remains unchanged.

There are typically two main shaders: a *vertex shader*, which computes each pixel's position, and a *fragment shader*, which calculates the pixel's final colour.

In computer graphics, rendering is typically done through a pipeline. In this pipeline, the vertex shader first transforms the vertex from the object's local space to the screen space. Then, the transformed vertex data is sent to the fragment shader, which applies colour to the pixel. To adjust the transformation, each of these shaders can obtain additional data such as matrices, positions, or colours.

Unity has its own high-level shaders written in HLSL (High-Level Shading Language) [18], which helps to abstract the low-level vertex and fragment shader programs.

One of them is called the Surface shader [19], which is used in this implementation. It eliminates a significant amount of boilerplate code, providing the user with a more comprehensive syntax, which the shader then converts to corresponding vertex and fragment shaders. The following sections will show the samples from Surface Shaders used in Project Tsunami.

4.2 Curving the Environment

Tsunami utilizes two surface shaders that are similar to each other in order to curve the virtual environment. While `TsunamiPlane.shader` performs transformation f_1, it is used only for objects such as planes with all vertices of a mesh having their global y-coordinate equal to 0. While it has limited use, it introduces no extra calculations necessary for a plane unlike `TsunamiComplex.shader`, which corresponds to f_2 transformation, additionally transforming the vertices above the ground.

Both shaders take 3 global values: `_Origin`, `_CurveDistance`, `_CurveCoefficient`, which are dynamically changed during the runtime in `CurvedWorld.cs`, which is the manager for curving the virtual environment storing and providing functions to modify these values.

Implementation of transformations f_1, covered in the subsection 2.2.1, can be seen below in Code 4.1, showing the code snippet from `PlaneShader.shader`. This section of the code is the same for both of the shaders.

```
1  function CurveFunction(vertexPos):
2      distX = abs(_Origin.x - vertexPos.x)
3      distZ = abs(_Origin.z - vertexPos.z)
4      vertexPos.y = pow(distX, _CurveExponent) *
           _CurveCoefficient
5      vertexPos.y += pow(distZ, _CurveExponent) *
           _CurveCoefficient
6      return vertexPos
7
8  function vert(v):
9      T_1 = multiply(unity_ObjectToWorld, v.vertex)
10     height = T_1.y
11
12     if IsTooFar(T_1):
13         return
14
15     T_2 = CurveFunction(T_1)
```

Code 4.1: `TsunamiPlane.shader`

Both shaders transform only vertices that are closer than the `_CurvedDistance` threshold. The `IsTooFar` function on line 12 is responsible for skipping the transformation of the current vertex if it is not within the threshold.

The auxiliary `CurveFunction` on line 1 represents a transformation f_1 taking a vertex as a parameter. It maps the points to the surface of a paraboloid, which is a parabola rotated around a y-axis, symmetric in x and z coordinates. That is the reason why the distances of the vertex from the `_Origin` can be calculated partially for x and z directions. The y-coordinate of the mapped point is calculated by the Equation 2.1 where the distances are plugged into d. The values are calculated for each direction respectively and added together, resulting in a point on the paraboloid.

In Figure 10, the plane is transformed using the `TsunamiPlane.shader` with a threshold value of `_CurveDistance`. This results in a paraboloid cut at the distance of the threshold.

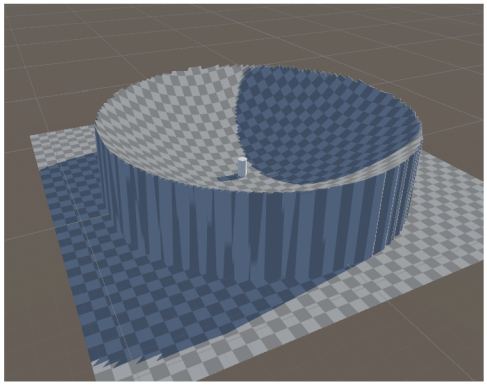

Figure 10: A plane transformed by the `TsunamiPlane.shader`

```
1  N_x.x += 1;
2  N_z.z += 1;
3  N_x = CurveFunction(N_x);
4  N_z = CurveFunction(N_z);
5
6  m_x = N_x - T_2;
7  m_z = N_z - T_2;
8
9  n = cross(m_z, m_x)
10
11 T_2 = T_2 + normalize(n) * height;
```

Code 4.2: `TsunamiComplex.shader`

The transformation f_2 covered in the subsection 2.2.2 is calculated by the `TsunamiComplex.shader`, which extends the shared functionality of `CurvedPlane.shader`. Neighbouring points N_x and N_z are calculated by adding a small value to the point's T_1 x and z coordinates, respectively. The f_1 transformation is then applied using the `CurveFunction`. Then, the vectors d m_x, m_z are constructed by subtracting the position of N_x and N_z from T_1 respectively. The normal vector n to the surface in the position of T_2 is calculated as a cross product of m_x and m_z. The point T_2 is then shifted by normalized vector n multiplied by the original height of the vertex T_1 on the input. Resulting in the final vertex above the surface of the paraboloid.

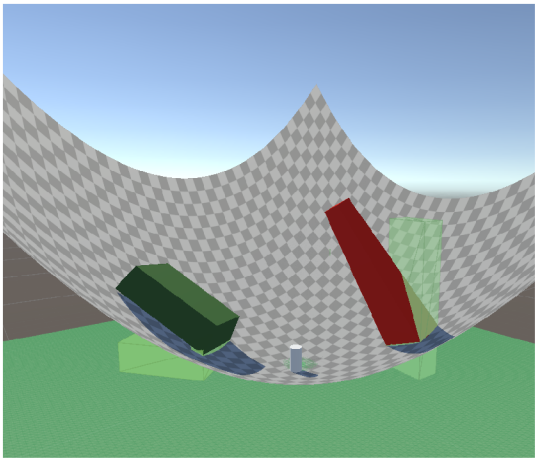

Figure 11: Screenshot of `TsunamiComplex.shader` from Unity editor

In Figure 11 objects above the plane transformed by the `TsunamiComplex.shader` can be seen where the green shapes represent the unaltered collision mesh of the curved objects.

4.3 Collisions with the Curved World

As described in the Design chapter chapter 2, to calculate the new position for the Tsunami teleport, the corresponding position must be first found in the flat world using the curved pointer ray as a naive inverse approximation of the curving function. This collision's coordi-

nates act as a new position after the teleportation sequence and are calculated within the `FlatWorldCollisionManager.cs` class.

It consists of two main and a few auxiliary functions. The first main function is called `FindCollsion` and is responsible for computing the necessary values for the second main function called `ShootCurvedRaycast`, which finds a collision of the curved pointer ray in the flat world. The code snippets from the `FlatWorldCollisionManager.cs` class will be presented and described. The theory covered in the section 2.3 will be presented in the form of a pseudo-code.

```
projectedRayOnXZ = CreateVector(controllerDirection.x,
    0, controllerDirection.z)
pointerSlopeDeg = Angle(controllerDirection,
    projectedRayOnXZ)
m = Tangent(ToRadians(pointerSlopeDeg))
if controllerDirection.y < 0:
    m = -m
```

Code 4.3: `FindCollision` code snippet 1/2

The first `FindCollsion` function snippet is shown in Code 4.3. It takes two parameters: the controller's global position and its forward-pointing vector, which are used to calculate the pointer ray's slope `m`. If the controller points to the ground, detected by the negative y-coordinate of its direction vector, the slope m is set as negative.

```
function ParabolaIntersectionDistance(m, k, c, positive):
    h = m / k
    result = 0

    if positive:
        result = (h + sqrt(h * h + 4 * (c / k))) / 2
    else:
        result = (h - sqrt(h * h + 4 * (c / k))) / 2

    return result
```

Code 4.4: `ParabolaIntersectionDistance` auxiliary function

The next step involves calculating the intersections of the pointer ray with the standard parabola at points P_1 and P_2. This is done using the `ParabolaIntersectionDistance` function shown in Code 4.4, which takes three parameters: the controller's slope m, `_CurveCoefficient` k, and the controller's y-coordinate c. Additionally, a `bool` value is

used to determine which of the two points to return, since coordinates of the intersections are calculated using the quadratic formula, where the equation for the pointer ray line and standard parabola are

$$y = m * x + c \tag{4.1}$$

$$y = k * x^2 \tag{4.2}$$

After setting these equations equal and substituting into the quadratic formula, an equation for x_1 and x_2 is obtained:

$$x_1, x_2 = \frac{\frac{m}{k} \pm \sqrt{(\frac{m}{k})^2 + 4\frac{c}{k}}}{2} \tag{4.3}$$

The calculations x_1, and x_2 are performed in two dimensions, resulting in the values being the **distances** of P_1 and P_2 from the controller.

```
function CalculatePointP(x, m, k, controllerPosition,
    projectedRayOnXZ, positive):
    y = m * x + controllerPosition.y
    positionOffset = x * normalize(projectedRayOnXZ)
    P = Vector3(controllerPosition.x, y,
        controllerPosition.z)
    P += positionOffset
    return P
```

Code 4.5: `CalculatePointP` auxiliary function

The P_1 and P_2 points can be created by taking the controller's x and z coordinates, shifting them along the normalized pointer ray's vector projected on the x-z plane (`projectedRayOnXZ`) multiplied by x_1 and x_2 acting as scalars, respectively. The y-coordinates of the P points are computed by substituting the x_1 and x_2 for x in the pointer ray's equation. The aforementioned calculations are performed in an auxiliary function `CalculatePointP` shown in Code 4.5.

```
dy1 = 2 * k * x1
dy2 = 2 * k * x2

alpha = abs((m - dy1) / (1 + m * dy1))
beta = abs((m - dy2) / (1 + m * dy2))
alpha *= TsunamiConstant
Q1 = Vector3(P1.x, 0f, P1.z)
Q2 = Vector3(P2.x, 0f, P2.z)
```

```

rayStartRatio = abs(x1) / (abs(x1) + abs(x2))
newPosition = ShootCurvedRaycast(Q1, Q2, alpha, beta,
    rayStartRatio)
return newPosition
```

Code 4.6: `FindCollision` code snippet 2/2

Finally, values α and β are calculated as angles between a pointer ray line and derivations in P_1 and P_2 respectively, where:

$$d_y = (k * x^2)\,' = 2k * x \tag{4.4}$$

an equation is obtained:

$$k_{1,2} = \left| \frac{m - d_y}{1 + md_y} \right| \tag{4.5}$$

Since it is a naive approximation of inverse function a `tsunamiConstant` is used to multiply the α by 1.5, which was manually found to tune the apex point of the Beziér curve.

The Q_1, Q_2 points are then constructed using the x-z coordinates of P_1 and P_2. The value `rayStartRatio` is calculated as a percentage portion of $x1$ to the sum of x_1 and x_2, which corresponds to the aerial distance between the controller (user) and point Q_1 (pointer ray). It is used to define what range of curve is to be calculated.

In order to find a new position during the teleportation sequence, a series of short rays are cast along a curved pointer ray calculated in `ShootCurvedRaycast` method. A set of three points Q_1, Q_2, Q_3 is used to create the curve by interpolating between them.

The first step is to calculate the point Q_3 using an auxiliary method called `CalculateIntersectionQ3`. This method takes previously computed angles α and β, creates lines p_1 and p_2 under these angles, oriented at each other from the points Q_1 and Q_2 and finds the intersection point Q_3. Below is a code snippet of the method.

```
1 Q1toQ2 = (Q2 - Q1).normalized;
2 Q2toQ1 = (Q1 - Q2).normalized;
3
4 p1 = new Vector3(Q1toQ2.x, alpha, Q1toQ2.z);
5 p2 = new Vector3(Q2toQ1.x, beta, Q2toQ1.z);
6
7 t = ((Q2.x - Q1.x) * p2.y - (Q2.y - Q1.y) * p2.x) /
      (p1.x * p2.y - p1.y * p2.x);
8 Q3 = Q1 + (p1 * t);
```

Code 4.7: `CalculateIntersectionQ3` auxiliary function

The coordinates of Q_3 are calculated using a formula derived from a system of equations of the lines p_1 and p_2. The lines in the parametric form are set equal and solved for the parameter t, obtaining the formula on line 7 in a code snippet from an auxiliary function `CalculateIntersectionQ3` shown in Code 4.7.

The curved pointer ray is then constructed by interpolating between the points Q_2, Q_3, and Q_1.

This interpolation is achieved using the `Lerp()` function, which is integrated into Unity. The function takes two points and an interpolation variable between 0 and 1, and returns a point that is linearly interpolated between the input points based on the value of the interpolation.

This creates a quadratic Beziér curve, as an interpolation of a line created by interpolation between the lines $[Q_2, Q_3]$ and $[Q_3, Q_1]$ within a for-cycle as seen in the Code 4.8 or in the Figure 8 covered in the Design chapter.

The `curveSegmets` is a variable defining a sampling number of the Beziér curve, and `i` is a for cycle step.

```
1  int i = curveSegments * rayStartRatio;
2
3  while i <= curveSegments;
4  {
5      interpolation = i / curveSegments;
6      Q1_3 = Lerp(Q1, Q3, interpolation);
7      Q3_2 = Lerp(Q3, Q2, interpolation);
8      nextCurvePoint = Lerp(Q1_3, Q3_2, interpolation);
9
10     ray.direction = (nextCurvePoint -
           ray.origin).normalized;
11
12     if (Raycast(ray):
13         return hitPoint
14     ray.origin = nextCurvePoint
15     i++
```

Code 4.8: `ShootCurvedRaycast`function sample code

In order for the ray casts to start from the position above the user, the initial for-cycle step `i` is calculated using `rayStartRatio` portion of `curveSegments`, resulting in the curve to start above the user. This is a crucial feature for a Project Tsunami to work in a scenario when the user is standing right next to a building pointing away, so the curve would not start inside a building.

Between every two points of the curve is cast a RayCast looking for collisions with the collision mesh of objects in a flat world. The first point of the curve is computed before the for-cycle and set as the `ray.origin`.

In each iteration of the for-cycle, a ray is cast from the `ray.origin` in the direction of the `nextCurvePoint` calculated on lines 5-8 in the Code 4.8. The new `ray.origin` is then set to the `nextCurvePoint` at the end of each iteration.

When the ray collides with a collision mesh of an object in a flat world, the function returns a RaycastHit object. If the angle from the y-axis of the hit surface normal is between 70-90 degrees, it is considered as a wall and disregarded. Otherwise, the position of the RaycastHit is set as a `_CurveOrigin` and a new position of the player ending the teleportation sequence.

The implementation of the Tsunami also contains a `delay` variable, which postpones the curving of the environment after pressing the trigger button, equal to the value of the variable in milliseconds. It is added in order to create an extra variation of Project Tsunami locomotion.

5 User Study

This chapter describes the design and results from a pilot user study conducted in order to compare two variations of Project Tsunami implementation with the baseline P&T method.

It was conducted in order to prove the functionality of the proposed implementation of Project Tsunami and to confirm if the alteration of the baseline P&T can speed up the navigation in an open virtual environment.

Additionally, it serves as a pilot study to compare two Project Tsunami variations to find the most effective version. Valuable insights are gained from assessing the strengths and weaknesses of each variation of Project Tsunami, which helps with future decision-making and optimizing the Tsunami locomotion technique.

5.1 Disclaimer

Our advisor decided that my colleague Vít Levinský and I, who are both doing a thesis on a similar topic, should make our studies compatible. This resulted in the sharing of the design of user study and mutual assistance during the study. However, it should be noted that the research, implementation of my locomotion methods, analysis of my results, and the text of this thesis were done by me.

5.2 Participants

The study included N=12 participants aged 19 to 24 years old, with a median of 23 years of age. Three participants were female, the rest were male, and all participants were university students.

Four participants stated they were using the VR for the first time, while the others said they used it occasionally. Only one participant reported having no gaming experience, while the rest were occasional (n=7), experienced (n=3), or expert players (n=1).

No participants had visual impairments not corrected by glasses/lenses, known balance disorders or were under any medication at the time.

The overall comfort level at the moment was measured on a scale of 1-10, 10 being very comfortable. The results averaged 8.3 while the median was 9; the lower scores were given almost entirely because of participants' fatigue when additionally asked.

5.3 Tested Locomotion Methods

The *baseline* method chosen for comparison was P&T since Project Tsunami is a modification of this technique. It was implemented using integrated XR interaction Toolkit components with a *Line Type* set to *projectile* in *XR Ray Interactor* component and a green semi-transparent cylinder reticle to mark the collision of the ray with suitable teleport surface. If the ray pointed to buildings or other forbidden surfaces, it changed the colour from white to red, and the reticle would be hidden.

The difference between the Tsunami variations used in the user study is that one starts to curve the environment instantly after pressing down the trigger button (*Tsunami*), while the other incorporates the delay after which the curving starts(*Tsunami with delay - TwD*). The delay was set to be 750ms and added with the intention of providing a user with a short time period to teleport on short distances when the immediate curvature of the environment is unnecessary.

Both variations marked the new teleport position with a green cylinder. The marker would not be shown if the user pointed to a surface unsuitable for teleport.

Presumptions about differences between the Tsunami and baseline were formulated into hypotheses. They aim to compare both baseline and Tsunami Methods as well as the variations of the Tsuami methods, and are presented below.

5.4 Hypotheses

The user study aimed to assess whether the set of formulated hypotheses could be confirmed:

H1 The Tsunami methods are more accurate at long-distance teleport than a baseline P&T.

H2 The Tsunami method allows the user to move faster in an open virtual environment of a city than using a baseline P&T.

H3 The use of Tsunami methods require fewer teleportation sequences than the baseline P&T to cover the same distance.

H4 The Tsunami methods are more enjoyable and preferred than the baseline.

H5 TwD is more accurate than Tsunami on short range.

H6 Tsunami is faster than the TwD.

5.5 Apparatus

The user study was conducted using an Oculus Quest 2 headset with Quest 2 controllers. The questionnaires were created within Google Forms and *testing application* in the Unity game engine.

5.5.1 The Testing Application

To compare the techniques, a testing application was created to test the accuracy and speed of the methods. The environment in the application is split into three parts: *Training Ground*, *Accuracy Test* and *Speed Test*. Each part is created in one scene and activated based on which part of the experiment is ongoing.

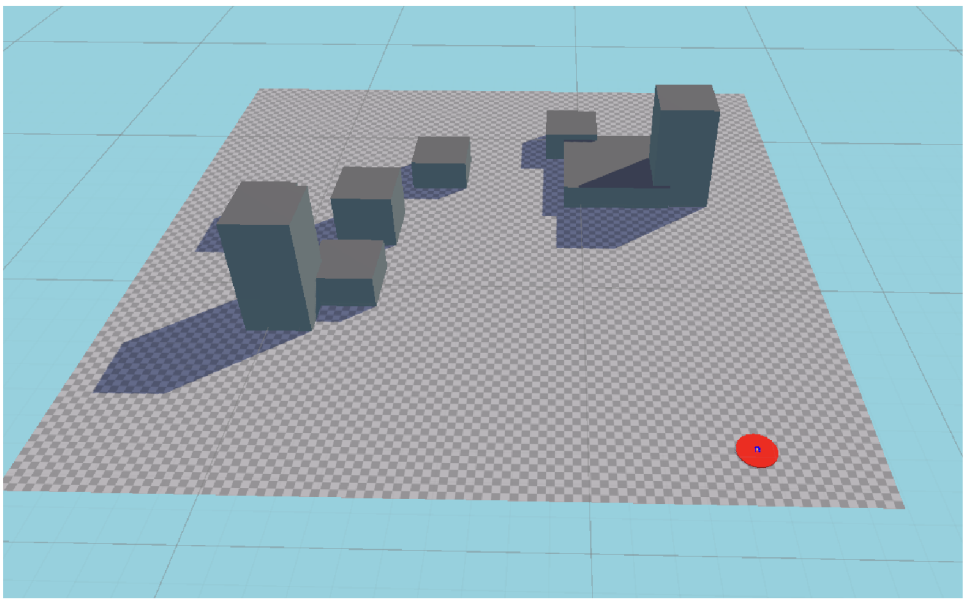

Figure 12: Screenshot of a Training ground

The **Training Ground** lets participants familiarize themselves with the tested technique in a demo world. They are able to move around the environment seen in Figure 12. The participant needs to teleport to a red circle to continue to an Accuracy Test.
The **Accuracy Test** starts with a dialogue window providing the participant one more time with introductory information about the test. The goal is to teleport as close as possible to the middle of the nine targets, gradually appearing one by one in front of the participant at three different distances: short = 30m, middle = 50m, and long = 100m. The player's distance from the centre of the target is then logged.

The test ends with a dialogue window informing the player about the end of the accuracy test, and by pressing *A*, the player is teleported to the speed test. The screenshots from the baseline P&T and Tsunami methods during the accuracy test are seen in Figure 13, showing the process of teleporting to one of the targets.

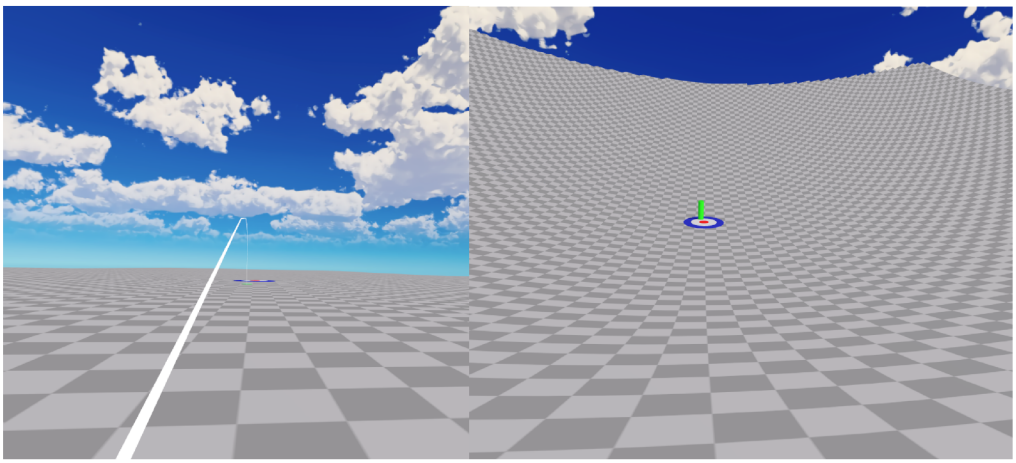

Figure 13: In-game screenshots of an accuracy test with a long-range target (left: baseline, right: Tsunami)

The **Speed test** starts similarly to the accuracy test with a dialogue window providing basic instructions. The goal is to complete a sequence of three paths consisting of six checkpoints each.

All paths begin in the same *starting position* on a virtual city square, and the timer starts when the participant teleports for the first time. The aerial distances between checkpoints were set at three values, each occurring twice: short = 87.5 m, middle = 175 m and long = 450m.

The checkpoints gradually appear one by one, with an arrow rendered in front of the player showing the direction to the next one. As the participant approaches each checkpoint, a sound is triggered to indicate that the checkpoint has been successfully visited. The previous checkpoint disappears, and the next one appears with an arrow pointing in its direction. The last checkpoint in each path is visually differentiated by a checkered texture depicting the finish and playing a different sound. After visiting the last checkpoint within one path, the participant is teleported to the starting position, and the timer is stopped. The screenshot from the test showing the arrow and checkpoint is seen in the Figure 14.

The time and position of each teleportation are logged, as well as a number of corrections and timestamps of checkpoint visits.

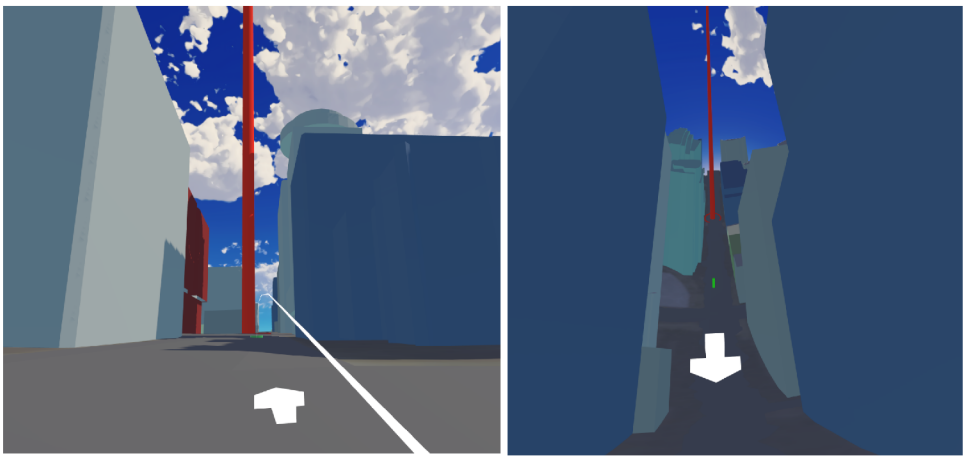

Figure 14: In-game screenshot of a speed test (left: baseline, right: Tsunami)

5.5.2 Questionnaires

A total of three types of questionnaires were created to collect participants' feedback in the form of qualitative data. All of them are included in the appendix of this thesis.

Consent form A.2 informs the participants about the purpose and risks of the user study and their rights within the procedure.

The **initial questionnaire** section A.3 contains questions to evaluate participants' demographic backgrounds and experiences in VR and gaming. They were also asked to state their comfort level on a scale of 1 to 10 in reference to potential motion sickness from a VR experience.

The **middle questionnaire** section A.4 handed out after completing the tasks within the testing application and consists of questions for intuitiveness and usability and open-ended questions for immediate feedback.

The **final questionnaire** Figure 20 handed out after testing all of the three methods and consists of mandatory open-ended and ranking

questions and an optional open-ended question for additional observations or comments about the user study.

5.5.3 Learning Effect

A Balanced Latin Square design was employed to minimise learning effects across trials. This method systematically varies the order of conditions for each participant, ensuring that each condition appears in each position across participants equally. The method was used to generate an order of the methods for each participant and the paths for each method.

Since the Balanced Latin Square creates a 2D array of 6 different orders of tested techniques, number 12 was chosen as the number of participants so that each specific order would get tested twice. The pre-defined orders for the first 6 participants are seen in the Table 1.

Table 1: Balanced LS for order of tested methods for participants P1-P6

Order / Participant	1.	2.	3.
1	A	B	C
2	A	C	B
3	C	A	B
4	C	B	A
5	B	C	A
6	B	A	C

5.6 Procedure

The user study was conducted in the Human-Computer-Interaction laboratory at the Faculty of Informatics of Masaryk University. It provides a big open space without any distracting elements, ensuring participants can focus on the task.

Upon arrival, the participant was briefed and asked to fill out the consent form and the initial questionnaire. Next, the participant was introduced to VR equipment and provided with sample videos to demonstrate the three locomotion techniques and controls. The study

administrator then selected an appropriate locomotion technique and instructed the participant to try it out within a training ground demo environment. After a few minutes of familiarising with the technique, the participant was asked to navigate to a red circle to begin the accuracy test. The participant was then instructed to complete the Accuracy and Speed test without further guidance from the study administrator. Once the Speed test was completed, the middle questionnaire was handed out, and the process was repeated for the other two methods. Finally, the participant received the final questionnaire, after which a short debrief followed, noting some additional comments. The user study lasted between 25 to 45 minutes.

5.7 Data Processing

Each user study experiment produced 4 files of logs. The accuracy test log, which contains 9 measurements for each user with the distance of the target from the player and the distance after teleport from its centre.

The speed test logged the number of checkpoint IDs and times when they were visited within each path, as well as the destination, time, and number of teleports after the teleport sequence.

The first step is to process the raw log files by transforming and organizing them into a format suitable for analysis. The data was always validated after each of the tests during the study to check whether the data were corrupted or not. The raw data of the same context were unified into single files and visualized in histograms, heatmaps, and boxplots for analysis.

As the majority of data collected from the user study does not follow a normal distribution or have uniform variance within the specific data groups, a non-parametric Friedman test was chosen to analyze the data. This test is suitable for data with different distributions and can be used in such cases to detect if there is a statistically significant difference between the groups.

The Wilcoxon signed-rank test was used as a post-hoc test with a Bonferroni correction for p-values to find whether there is a statistically significant difference between some of the groups. Additionally,

effect size r. Within the accuracy test, some of the participants accidentally pressed a trigger button, resulting in faulty measures, which the participants reported to the user study administrator and were externally noted. The marked outliers were then replaced with the mean of the affected data column.

There was participant P8, who did not notice the arrow pointing to the next checkpoint in the speed test, resulting in an abnormally long time for each path. Since these measurements had different conditions for the experiment, they were discarded.

Additionally, during the speed test, the Oculus headset lost track of the space a few times due to participants' rapid head movement, which interrupted the experiment by the length of time needed to set it up again. These were always immediately reported to the user study administrator and noted. It happened once for baseline, twice for TwD, and 3 times for Tsunami. These anomalies were then replaced by a mean calculated based on the affected method and path.

The user study environment and one of the participants are seen in Figure 15.

Figure 15: A participant in the middle of the user study

6 Results

In this chapter, the results obtained from the research will be presented in detail. The data collected during the study will be subjected to statistical analysis to test the thesis hypotheses. The statistical analysis will be aimed at identifying any statistically significant differences in the data.

6.1 Accuracy Test

The Tsunami methods show lower precision over a long range than the baseline method. The results prompt a closer look into the factors influencing participant performance and method efficacy.

Figure 16 shows the accuracy results in the form of a box plot. It provides a visualization of the interquartile range (white box), mean (red line) and spread of distances (whiskers) across the evaluated methods.

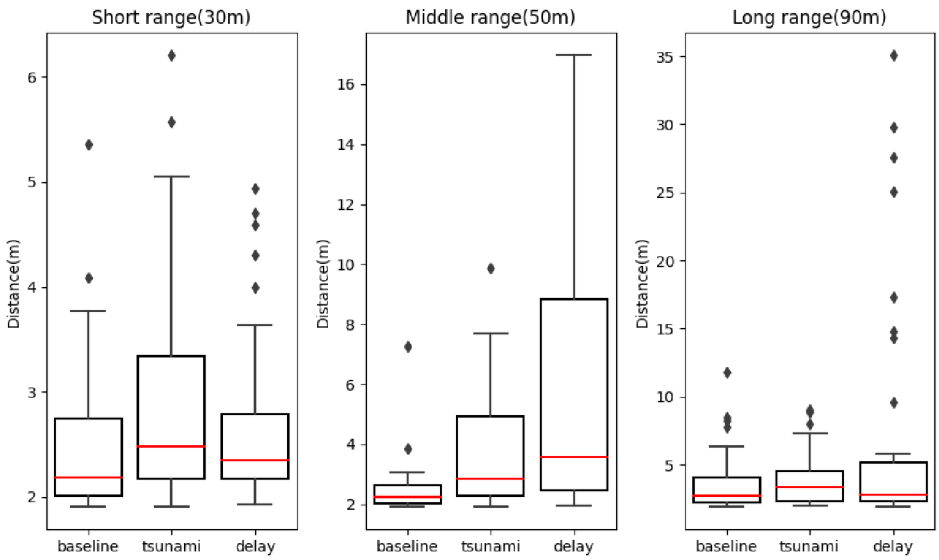

Figure 16: Box plot for each of the tested accuracy ranges

Since the faulty measurements marked as outliers caused by accidental teleports were replaced with mean values in the preprocessing

of the data, the black dots in the boxplots represent values outside the interquartile range and are not considered outliers. More on the spread of the values is covered in the section 7.1.

Table 2: Median distance from the target

	Baseline	Tsunami	TwD
30m	**2.47**	3.75	2.72
50m	**2.98**	3.84	5.81
90m	**3.63**	3.95	7.13

The table Table 2 shows the median values for each tested range. The baseline resulted in the lowest median distance for all three ranges.

Friedman test evaluated for each range found statistically significant differences (SSD) between methods' accuracy at short and mid-range (short:$p=0.046$, mid:$p<0.001$) and no SSD at long ($p=0.125$).

When using post-hoc Wilcoxon signed rank test with Bonferroni p-value correction and effect size r, only two SSDs were found. Both were at mid-range between baseline-Tsunami ($p=0.0015$, $r=0.142$) and baseline-TwD ($p<0.0001$, $r=0.177$).

This finding rejects the first hypothesis **H1**, saying that the Tsunami methods are more accurate at long-distance teleport. While the baseline performs significantly better at mid-range than both of the Tsunami variations, no SSD was found at long-range between baseline and any of the Tsunami methods. This means that the Tsunami methods can be considered equally accurate on short and long-range teleport as the baseline, but no conclusion about overall accuracy can be made because of the baseline's SSD on mid-range.

No significant difference was found between the accuracy of Tsunami and TwD. Although the median distance of TwD from a target was lower at a short distance than Tsunami, it is not enough to confirm the hypothesis **H5** that TwD is more accurate at a short distance than Tsunami.

6.2 Speed Test

Tsunami methods had both shorter mean times to complete the path and a lower count of needed teleportation sequences to reach the final checkpoint. These results are discussed and visualized below.

Table 3: Median time for each paths P1-3 in speed test (s)

	Baseline	Tsunami	TwD
P1	73.16	**52.07**	62.23
P2	50.66	**39.47**	52.14
P3	**44.82**	52.36	78.54

The median times of paths from the speed test are shown in the Table 3. While Tsunami has the lowest median times for P1 and P2, the baseline was the fastest technique on P3.

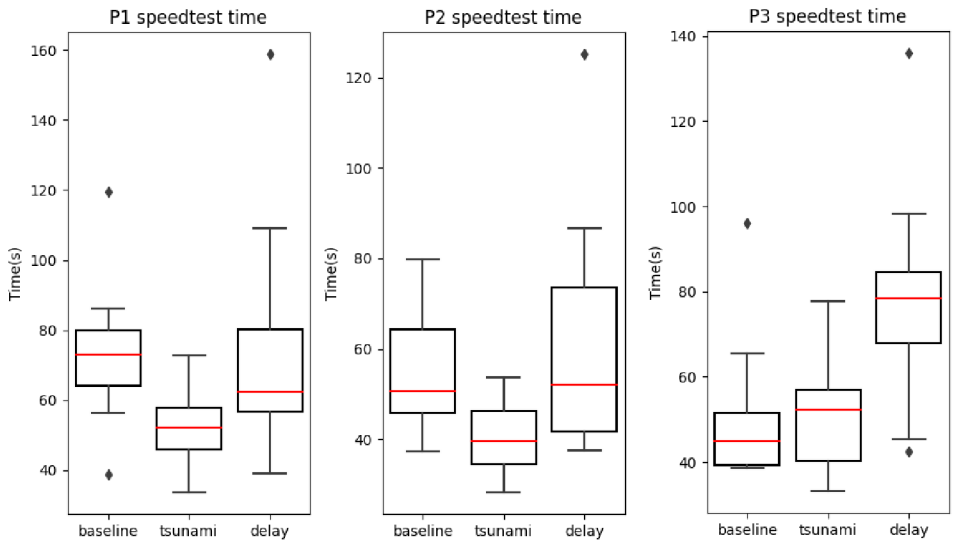

Figure 17: Speed Test times box plots

The results regarding the time spent completing the paths are shown in Figure 17. An SSD was found between the methods for P1 (p=0.0117) and P3 (p=0.004). The Wilcoxon post-hoc test did not

find any SSD between the methods for P1, but two were found for P2 between Baseline-TwD (p=0.001, r=0.41) and Tsunami-TwD (p=0.007, r=0.39).

Even though these results and findings mostly support the second hypothesis **H2**, that Tsunami methods are faster for navigating through the open virtual environment, it cannot be proved, and therefore, the results are inconclusive.

Similarly, the hypothesis **H6** stating that Tsunami is faster than TwD cannot be fully confirmed or rejected. Though the Tsunami method was faster on each path compared to TwD, with a difference of at least 10 seconds based on median values, there was no significant difference between these methods on any of the paths (p=0.147). Hence, the results only partially support the better efficiency of the Tsunami.

Table 4: Median number of teleports in the Speed Test for paths P1-3

	Baseline	Tsunami	Tsunami /w delay
P1	59.9	**22.4**	37.1
P2	67.1	**18.3**	21.9
P3	49.0	**25.2**	34.5

6.2.1 Teleport Count

The mean number of teleports for each of the paths P1-P3 is shown in Table 4. Tsunami was a technique with the fewest teleports needed to complete any of the three paths within the speed test, while the baseline needed almost twice as many teleports as the Tsunami methods.

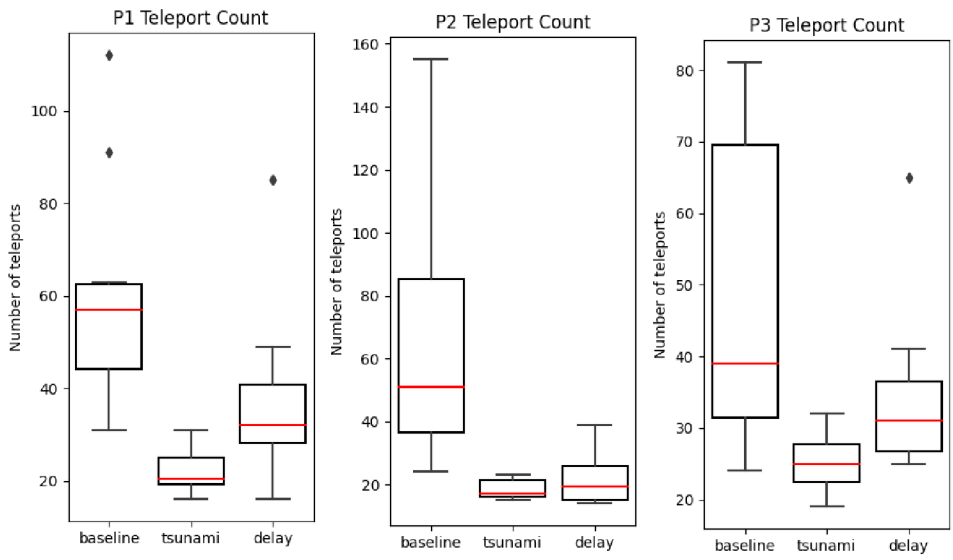

Figure 18: Number of teleports for each path in Speed Test

Looking at the boxplot with a number of teleports for each path in Figure 18, the outliers in the baseline and TwD were caused by one of the participants (P3) utilising a "spamming" technique by teleporting more frequently at mid to short distances. Even though this technique was second fastest within the baseline method, the times were below average within the Tsunami methods.

Table 5: Teleport count p-values and effect size *r*

	P1	P2	P3
baseline-Tsunami	**p=0.002**, r=0.70	**p=0.002**, r=0.54	**p=0.002**, r=0.49
baseline-TwD	$p > 0.05$	**p=0.004**, r=0.49	$p > 0.05$
Tsunami-TwD	**p=0.01**, r=0.34	$p > 0.05$	**p=0.014**, r=0.35

Using the Friedman test, an SSD between the methods was found on each path. (P1: p=0.0017, P2: p=0.0013, P3: p=0.002). The individ-

ual p-values with effect sizes r calculated by Wilcoxon signed rank post-hoc test with Bonferroni p-value correction are shown in Table 5.

As seen in the Table 5, the SSD was found on each path between the baseline and Tsunami. Even though the TwD was not significantly different compared to the baseline on P1 and P2, it still required fewer teleportation sequences than the baseline on all of the paths. This finding confirms the second hypothesis **H2** that the Tsunami methods require fewer teleports to cover the same distance.

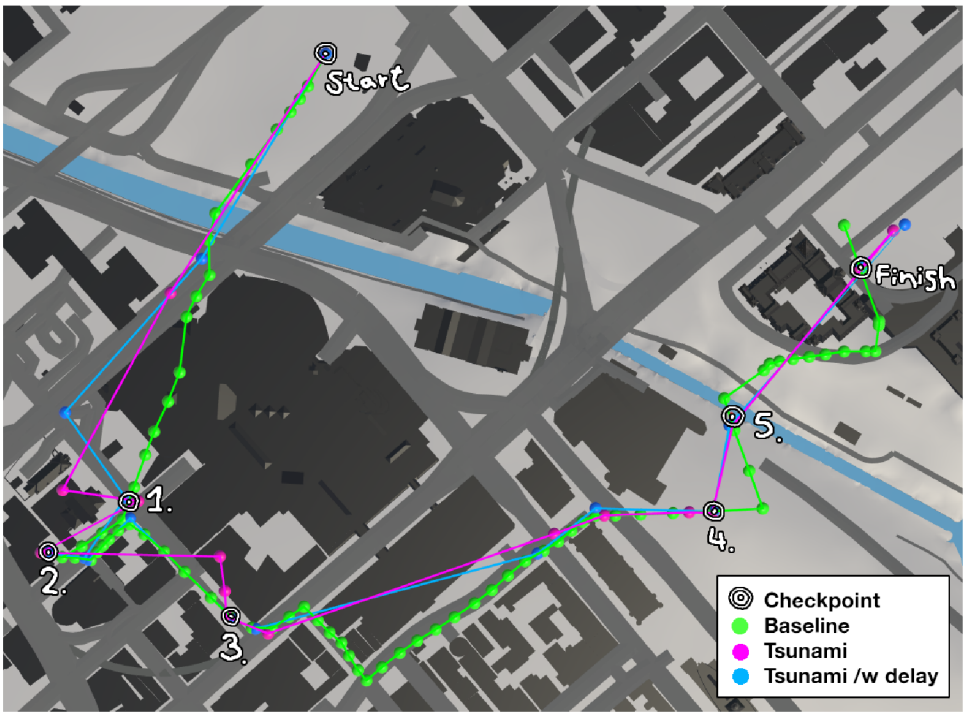

Figure 19: Visualised paths for each method in one run

This phenomenon is also seen when visualising the methods in one of the participant's runs in the Figure 19. Looking at the teleports between the fifth checkpoint and finish, the baseline teleport was used 14 times compared to two teleports for each of the Tsunami methods. The much higher frequency of teleportation sequences is seen as the number of dots representing one teleport.

6.3 User Feedback

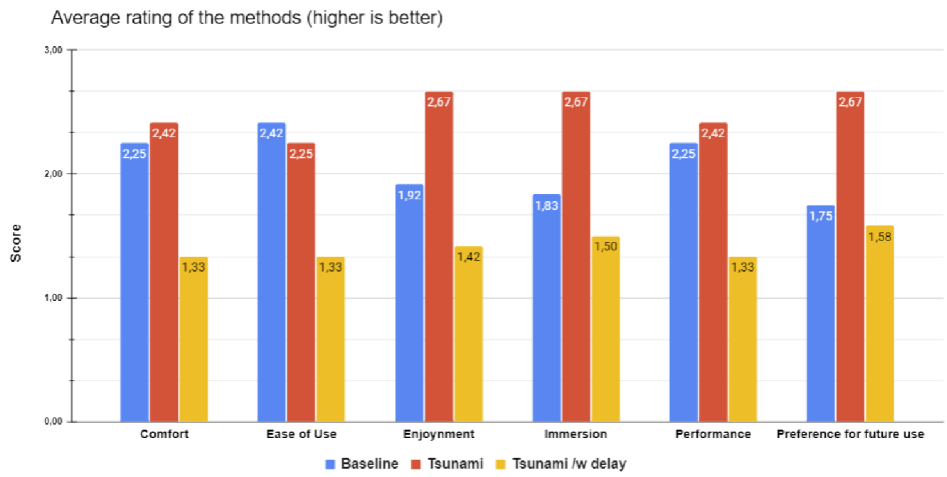

Figure 20: Evaluation of ranking questions from the final questionnaire

The scores were calculated using a weighted sum where the method in the first place was worth 3 points, the second 2 and the last 1. The sum was then divided by the number of participants, resulting in a normalized score.

The SSD was set as the difference between the scores exceeding the value 1. By this definition, the statistically significant differences were mostly between the Tsunami methods, favouring the Tsunami without delay in each case. The results are seen in the Figure 20.

The SSD was found between the Tsunami methods in all of the measured aspects except the *Ease of use*, where the significant difference was between the baseline and TwD, in favour of the baseline.

Although the difference in scores for enjoyment and preference for future use between the baseline and Tsunami is below the SSD threshold, Tsunami still scored significantly higher in both aspects. While this finding cannot fully confirm hypothesis **H4**, that the Tsunami methods are more preferred and enjoyable, it partially supports it and cannot be rejected either.

The results of participants' preference of each method on the individual distance are seen in the Figure 21. Almost all of the participants preferred the baseline for teleportation at a short distance. However, the Tsunami was the most preferred mid- and long-range teleports method.

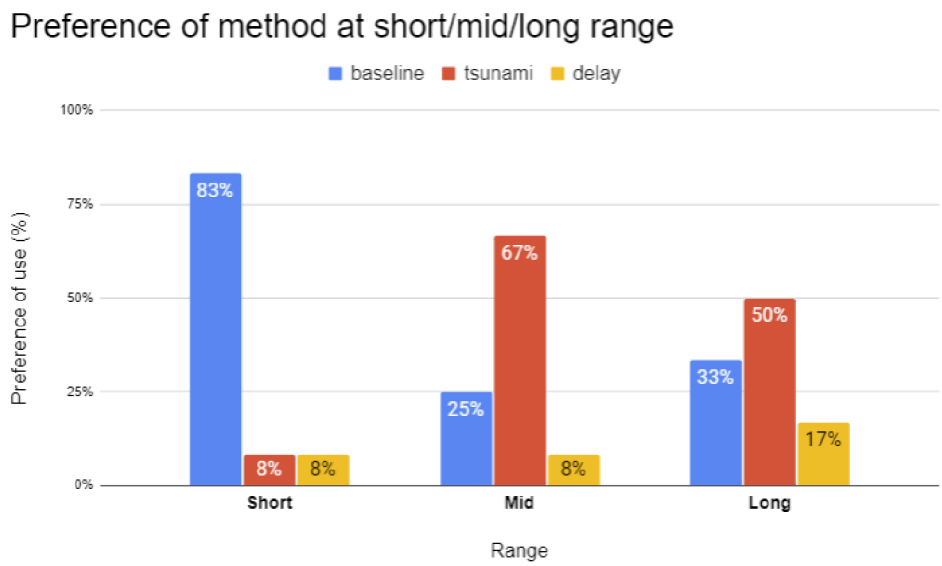

Figure 21: Preference of the methods at short/mid/long range

Participants were also asked to rate the ease of use of and whether they had a sufficient level of control of the method on a scale of 1 to 4, with 1 being strongly disagree and 4 being strongly agree. Regarding the ease of movement, participants were asked how much they agreed with the statements: *I was able to move easily in space using this method*. Results are seen in the Figure 22.

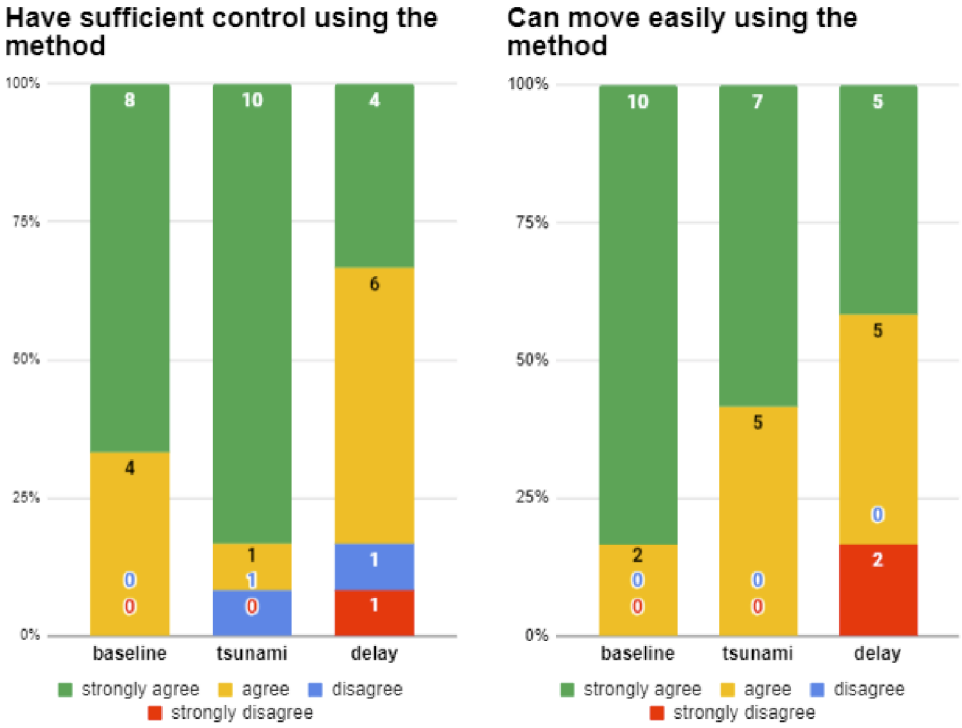

Figure 22: Evaluation of Likert questions from the middle questionnaire

Regarding the *sufficiency of control*, the tsunami was rated best and TwD worst. The delay caused some of the participants a little frustration, resulting in the answers being most disagreeing. The baseline was rated a bit worse than Tsunami since some participants reported not seeing where they were teleporting when pointing behind buildings.

The *ease of movement* results for the Tsunami methods were primarily affected by implementation limitations (as described in Limitations section 8) as well as frustration from delays. Some participants experienced bugs during the testing, which influenced their ratings to be slightly lower.

7 Discussion

7.1 Accuracy

The values are considered legitimate when looking at the spread of values in the long-range accuracy boxplot in Figure 16. This is because, as shown in Figure 13 at long-range in the case of baseline, the target looks almost like a 2D line and can be challenging to aim at the centre and the fact that the baseline was equally accurate as the Tsunami on long-range was unexpected.

In the case of the Tsunami methods, the wide range of distances can be explained by the curvature of the environment and the fact that some of the participants did not wait for the curving to stop. Some participants have even reported experiencing this observation.

This phenomenon becomes more noticeable when looking at the extreme values of the TwD in the Figure 16. During the delay period, participants aimed at the target, but the curvature started the moment before releasing the button, leading to poor results.

The same problem is observed at the TwD's mid-range boxplot in the Figure 16 where it occurred even more frequently, resulting in a wide distribution of data not marked as outliers, unlike in the long-range.

It seems that the same factor was likely responsible for the wider range of values observed for the Tsunami method at short range in the Figure 16. This finding may explain why the Tsunami method had a higher median value in the Table 2 than the TwD on short-range.

7.2 Speed

The longer times for the TwD method in the Table 3 were most probably caused by the delay. This can be naively checked when we take an average of the median counts of teleports for TwD for each of the paths and multiply it by the delay length of 0.75 seconds. That is equal to approximately 23 additional seconds spent waiting for the delay, and the average difference between the median of Tsunami and TwD time is around 16 seconds. The difference between these times is 7 seconds, but if only 75% of the teleports were considered to be on

long-range (every 4th teleport is on short-range), the same calculation would result in a difference of only approximately 1.5 seconds, which supports the belief that delay caused the longer times.

Looking at the Figure 17, the baseline results for path 3 were not expected, where the method even had the best median time on this path. It is interpreted that the cause for this finding would be the placement of path P3 within the city and one of its checkpoints in a narrow alleyway. The other two paths contained parts that were much more open and were not obstructed by high buildings.

The path P3 was almost entirely between the dense high buildings, which limited the use of the curving Tsunami methods and was also reported by some of the participants. The checkpoint in the alleyway caused some participants to wander around while they could not find it. Similarly, it was reported by four participants, that it was easier to navigate through these dense locations with baseline rather than Tsunami methods.

7.3 Teleport Count

The reason behind the difference in teleport counts between the Tsunami and TwD in the Figure 18 is interpreted as the participants being more prone to teleport on shorter distances within the delay time period. This was the delay feature's aim and was even arbitrarily mentioned by two participants.

Participant P6, who was the only participant experienced with VR said: *"I have a feeling that this method (TwD) was a combination of the previous ones, intended for me to be able to combine short + long distances... Well, when using it on short distances, it often bugged, so I preferred not to use it; and for long distances, I had to wait for the delay, for which I no longer had a use for"*. The mentioned bug is covered in Limitations 8.

Participant P4 said: *"I figured the delay was added in order to teleport on the short distances and quite liked it, but the length of the delay should be shortened."*

7.4 Motion sickness

Three participants reported experiencing subtle symptoms of dizziness related to motion sickness during the user study. The first participant experienced the symptoms at the end of the test, while the second one felt it throughout the entire study. The third participant experienced the symptoms after spending almost 10 minutes locating the next checkpoint in the Speed Test. However, the symptoms disappeared in this case before testing the next method. None of the participants who experienced symptoms reported any performance issues when asked.

7.5 Suggestions

Some of the participants suggested adding a visual cue indicating that the current teleport is not possible by showing a red x symbol in front of the player and curving the environment in a manner different from linear. These are very interesting points and are suggested to be tried and implemented in future work.

Similarly, the suggestion by participant P4 to shorten the delay was made by 5 other participants, indicating a possibility for future work to find the ideal delay length for the TwD locomotion technique.

8 Limitations

In order to make the Tsunami methods and the baseline P&T comparable in functionality, the distance of the environment curving was limited. This limit was set to be equal to the farthest range that the baseline can travel through teleportation. Without this limitation, the Tsunami methods could teleport through half of the map within one teleportation sequence, making it invalid to compare with the baseline.

Regarding the mentioned *bugs* from participant P6. There are edge cases in the `ShootCurvedRaycast` function when the collision is either not found, or the pointer ray does not collide with the standard parabola when pointing straight down (very close range) or up. In these cases, the function returns a predefined value outside the virtual city, and the player is teleported to this position. A button *B* on the controller was utilised to return the player to the previous position and log this correction. This was done in order for the teleport to work with each press of the trigger button. If the participant pressed teleport again, when outside the city, the correction would not work, and the test would have to be restarted.

This happened to 4 participants, each within the first few teleports in the speed test, so it did not influence the results, and the experiment needed to be restarted. One participant then reported that it forced them to be more careful.

This is acknowledged to be a limitation and is suggested to be fixed in future work.

9 Conclusion

The aim of this thesis was to develop Project Tsunami, a new VR locomotion technique that utilizes the curvature of the virtual environment to teleport to previously inaccessible locations. Project Tsunami is a modification of the traditional Point and Teleport method and aims to make locomotion more efficient in an open VR environment. To compare the effectiveness of two variations of Project Tsunami with the baseline Point and Teleport method, a pilot user study was conducted with 12 participants.

The results of the study showed that the baseline method was more accurate than both Tsunami variations at mid-range teleportation. However, there were no significant differences in accuracy between the three methods at short and long-range teleportation. In terms of time efficiency, the Tsunami methods required significantly fewer teleports than the baseline. The Tsunami method without delay had shorter median completion times, indicating better efficiency presumption. However, there was no significant difference in time efficiency compared to the baseline.

Participants preferred the Tsunami methods for mid- and long-range teleportation, while the baseline was favoured for short distances. Although there were some limitations and frustrations, participants found Tsunami methods more enjoyable and preferable for future use. Additionally, minimal to no motion sickness symptoms were triggered when using the Tsunami Methods.

The comparison of the Tsunami methods revealed that Tsunami without delay is a more viable option for future use in terms of both performance and preference. However, feedback from participants on the delay in the Tsunami with delay prompts an option for future work to find an ideal delay length.

Overall, this thesis contributes to the VR locomotion research field through the development of Tsunami, a novel VR locomotion technique which proved to be significantly more efficient in the number of teleport sequences needed to cover the same distance as the baseline. Although the pilot study provides valuable insights, further research and improvements in implementation are necessary to draw more definitive conclusions in terms of the Tsunami's time efficiency and

accuracy. This simultaneously prompts another opportunity for future work to find a more effective and accurate approach to finding collisions in the curved virtual environment.

Bibliography

1. BOZGEYIKLI, Evren; RAIJ, Andrew; KATKOORI, Srinivas; DUBEY, Rajiv. Point & Teleport Locomotion Technique for Virtual Reality. In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. Austin, Texas, USA: Association for Computing Machinery, 2016, pp. 205–216. CHI PLAY '16. ISBN 9781450344562. Available from DOI: 10.1145/2967934.2968105.
2. BOLETSIS, Costas. The New Era of Virtual Reality Locomotion: A Systematic Literature Review of Techniques and a Proposed Typology. *Multimodal Technologies and Interaction*. 2017, vol. 1, no. 4. ISSN 2414-4088. Available from DOI: 10.3390/mti1040024.
3. BOLETSIS, Costas; CEDERGREN, Jarl Erik. *VR locomotion in the New Era of Virtual Reality: An empirical comparison of prevalent techniques*. Hindawi, 2019. Available also from: https://www.hindawi.com/journals/ahci/2019/7420781/.
4. PRITHUL, Aniruddha; ADHANOM, Isayas Berhe; FOLMER, Eelke. *Teleportation in virtual reality; a Mini-Review*. Frontiers, 2021. Available also from: https://www.frontiersin.org/articles/10.3389/frvir.2021.730792/full.
5. LÍNDAL, Páll J.; JÓHANNSDÓTTIR, Kamilla Rún; KRISTJÁNSSON, Unnar; LENSING, Nina; STÜHMEIER, Anna; WOHLAN, Annika; VILHJÁLMSSON, Hannes Högni. Comparison of Teleportation and Fixed Track Driving in VR. In: *2018 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. 2018, pp. 1–7. Available from DOI: 10.1109/VS-Games.2018.8493414.
6. TECHNOLOGIES, Unity. *Unity Engine Game object documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/2021.3/Documentation/Manual/GameObjects.html.
7. TECHNOLOGIES, Unity. *Unity Engine Component description* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/ScriptReference/Component.html.

8. TECHNOLOGIES, Unity. *Unity Engine Scene description* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/Manual/CreatingScenes.html.

9. TECHNOLOGIES, Unity. *Unity Engine Transform component documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/ScriptReference/Transform.html.

10. TECHNOLOGIES, Unity. *Unity Engine Script documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/ScriptReference/Component.html.

11. TECHNOLOGIES, Unity. *Unity Engine Mesh collider documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/2021.3/Documentation/Manual/class-MeshCollider.html.

12. TECHNOLOGIES, Unity. *Unity Engine Physics class documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/2021.3/Documentation/ScriptReference/Physics.html.

13. TECHNOLOGIES, Unity. *Unity Engine Ray documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/ScriptReference/Ray.html.

14. TECHNOLOGIES, Unity. *Unity Engine Physics.RayCast documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html.

15. TECHNOLOGIES, Unity. *Unity Engine RaycastHit object documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/ScriptReference/RaycastHit.html.

16. TECHNOLOGIES, Unity. *Unity Engine XR Interaction toolkit description* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.5/manual/index.html.

17. TECHNOLOGIES, Unity. *Unity Engine OpenXR plugin manual* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.8/manual/index.html.

18. TECHNOLOGIES, Unity. *Unity Engine HLSL documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/Manual/SL-ShaderPrograms.html.

19. TECHNOLOGIES, Unity. *Unity Engine Surface shader documentation* [online]. [visited on 2023-10-02]. Available from: https://docs.unity3d.com/Manual/SL-SurfaceShaders.html.

A An appendix

A.1 Appended Files

The IS MU archive contains the following files:

- The Project Tsunami Unity project - /ProjectTsunami.zip
- Accuracy test logs in /logs/Accuracy.csv
- Speed test logs - checkpoint times in /data/Checkpoints.csv
- Speed test logs - teleport records in /data/Teleports.json
- Jupyter notebook for analysis of accuracy results in /jupyters/Accuracy.ipynb
- Jupyter notebook for analysis of checkpoint times in /jupyters/Checkpoints.ipynb
- Jupyter notebook for analysis of teleports in /jupyters/Teleports.ipynb
- Questionnaire results and analysis in /data/Questionnaires.xlsx

A.2 Consent form

The consent form, including all questionnaires, is available here.

A.3 Initial Questionnaire

1. Participant ID
2. Age
3. Gender
4. Occupation
5. How would you rate your overall experience with virtual reality?
6. How often do you engage in activities that require the use of virtual reality headsets?
7. How would you describe your level of experience with video games in general?

8. Do you experience discomfort or motion sickness while using VR or playing video games? If yes, please describe.
9. Do you have any visual impairments that are not corrected by glasses or contact lenses?
10. Do you have any known balance disorders?
11. Are you currently under medication that may affect your balance or spatial orientation?
12. On a scale from 1 to 10, how would you rate your current comfort level?
13. Are you familiar with any locomotion (movement) techniques in virtual reality? If yes, please list them.

A.4 Middle Questionnaire

1. Did you find the locomotion method intuitive?
2. Did you feel that you had sufficient control over your movement in the VR environment with this method?
3. Did you experience any discomfort or motion sickness while using any of the locomotion methods?
4. How do you agree with this statement? I was able to move easily in space using this method.
5. Did you notice any lag or delay in response when using the locomotion method?
6. Please describe any specific issues you encountered while using the locomotion method.
7. Was there anything about the locomotion method that made it difficult to focus on the task at hand?
8. Do you have any suggestions for how the locomotion method could be improved?
9. Is there anything else about your experience with the locomotion method that you would like to share?

A.5 Final Questionnaire

1. Which locomotion method did you prefer for the short/middle/long-range locomotion, and why?
2. How would you describe your overall experience with the VR locomotion methods used today?
3. Rank the following locomotion methods based on how comfortable they were for you.
4. Rank the locomotion methods from easiest to use to the most challenging.
5. Rank the locomotion methods based on the level of enjoyment you experienced.
6. Rank the methods according to the level of immersion they provided.
7. Rank the locomotion methods based on how well you felt your performance was.
8. Rank the locomotion methods based on your preference for their use in future VR applications.
9. Did you notice any delayed effects (such as motion sickness, fatigue, etc.) after using any of the locomotion methods?
10. Do you have any suggestions for how the locomotion methods could be improved for a better user experience?