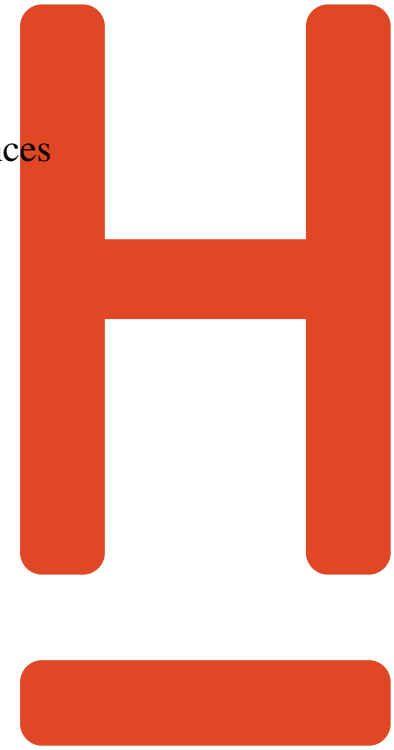

# Motion Sickness Reduction for 6-DoF-Navigation in a Virtual Solar System

Moritz Zeumer

Master Thesis

Hochschule Hannover
Faculty IV – Business and Computer Science
Course of Studies M. Sc. Applied Computer Sciences

April 29, 2021



**Author**

Moritz Zeumer
Matrikelnummer: 1498947
E-Mail: M_Zeumer@gmx.de

**Examiner**

Prof. Dr. Volker Ahlers
Hochschule Hannover
Faculty IV, Computer Science
Ricklinger Stadtweg 120
30459 Hannover

**Second Examiner**

M. Sc. Simon Schneegans
German Aerospace Center (DLR)
Institute for Software Technology
Software for Space Systems and Interactive Visualization
Lilienthalplatz 7
38108 Braunschweig

**Declaration of Authorship**

I hereby declare that this thesis, and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that:

1. Where I have consulted the published work of others, this is always clearly attributed.
2. Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my own work.
3. I have acknowledged all main sources of help.
4. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Hannover, April 29, 2021
Location and Date Signature

# Contents

# 1 Introduction

## 1.1 Virtual Reality

## 1.2 CosmoScout VR

## 1.3 Scope of the project

# 2 Background and Related Work

While Virtual Reality technology has gained more and more traction over the recent years, 30% to 80% of users encounter some form of sickness symptoms during their exposure to virtual environments [1]. Additionally, these sickness symptoms can have lasting effects after the exposure as well [2]. The high number of affected users has led to cybersickness being one of, if not the biggest roadblock to a more widespread adoption of Virtual Reality Devices.

According to LaViola [2] the symptoms of exposure to virtual environments include:

- Eye strain
- Headache
- Pallor
- Sweating
- Dryness of mouth
- Fullness of stomach
- Disorientation
- Vertigo
- Nausea
- Vomiting.

Vertigo, in the case of VR-sickness particularly benign paroxysmal positional vertigo (BPPV), is a condition where the individual experiences a false sense of motion, or spinning, and objects or surroundings appear to swirl or move [3].
Several studies also found that severity of symptoms increases with longer exposure times to virtual environments [4, 5, 6]. However, some studies show that users can adapt, and overall sickness reduces with repeated exposure [7].

Throughout the study of these symptoms, several terms have been used to compound these sickness symptoms that appear to be similar to the symptoms of motion sickness.
Initially, the term Simulator Sickness was used to describe motion sickness encountered during exposure to flight simulators [8]. The term originated from the assessment of military flight simulators [9]. While Simulator Sickness is still used in recent publications, the terms Cybersickness or VR Sickness are generally used to differentiate, and closer examine the side effects of virtual environments from simulator sickness [8, 10].
The term VR Sickness specifically is used in discussions and studies about sickness symptoms

involving head-mounted displays (HMD) [11, 12]. This terminology is often used interchangeably across literature.
The terms Cybersickness and VR Sickness will be used in this study, as Stanney, Kennedy, and Drexler [13] argue that, while sickness from virtual environments shares many of the symptoms also experienced during simulator sickness or motion sickness, the sickness profiles are different.

| | **Simulator sickness** | **Sea sickness** | **Space sickness** | **Cybersickness** |
|---|---|---|---|---|
| Highest rating | Oculomotor | Nauseagenic | Nauseagenic | Disorientation |
| Middle rating | Nauseagenic | Oculomotor | Disorientation | Nauseagenic |
| Lowest rating | Disorientation | Disorientation | Oculomotor | Oculomotor |

Table 2.1: Related conditions symptom profiles according to Rebenitsch and Owen [1].

According to Rebenitsch and Owen [1] cybersickness and other sickness symptoms similar to motion sickness are polysymptomatic (many symptoms) and polygenic (different manifestation for individuals) and therefore complex to understand and describe. To make the sickness and its symptoms easier to survey and examine, Kennedy et al. [9] categorize the symptoms listed above into three categories:

- Nauseagenic symptoms (dryness of mouth, fullness of stomach, nausea, etc.)
- Oculomotor sypmtoms (eye strain, headache, etc.)
- Disorientation symptoms (vertigo, dizziness, etc.)

The main arguments for the distinction between simulator sickness and cybersickness are that during cybersickness, disorientation symptoms rank highest and oculomotor symptoms rank lowest, while simulator sickness and traditional motion sickness usually have the inverted profile, where disorientation symptoms rank lowest [13].

Cybersickness can also occur without stimulation to the vestibular system, purely through visual cues, unlike motion and simulator sickness, where stimulation of the vestibular system is needed, but not visual stimulation [2]. Additionally, Stanney et al. [13] determined that cybersickness can be up to three times more severe than simulator sickness. Saredakis et al. [8] also note significantly higher average Simulator Sickness Questionnaire scores, although both mention, the scores and questionnaire were established with a focus on military flight simulators used by military personnel. While recently, the Simulator Sickness Questionnaire has been adopted to measure cybersickness in virtual environments, which might be the reason for the higher average scores [8].

## 2.1 Common causes of cybersickness

Over the recent years there have been several theories trying to explain the sickness symptoms experienced during extended exposure to virtual environments, especially since the commercialisation of head-mounded virtual reality devices.
The most common Theories are the sensory conflict theory and the postural instability theory. Additionally, there are some theories that try to explain why sickness symptoms occur in virtual environments like the rest frame theory, and the vergence accommodation conflict theory.

### 2.1.1 Sensory conflict theory

The generally most accepted, and widespread theory is based on a sensory mismatch either between sensory systems of the body, or between sensory input and expectation given the perceived environment. Most commonly, a sensory conflict due to vection (the illusion of self movement while stationary) is argued to be the main cause of cybersickness [14, 15]. Although, other studies like Palmisano, Mursic, and Kim [16] suggest, that vection is neither the sole, nor primary source of sensory conflict. Sensory conflicts like vection can also occur outside virtual environments, for example when a person is in a stationary vehicle while an adjacent vehicle begins to move [2].

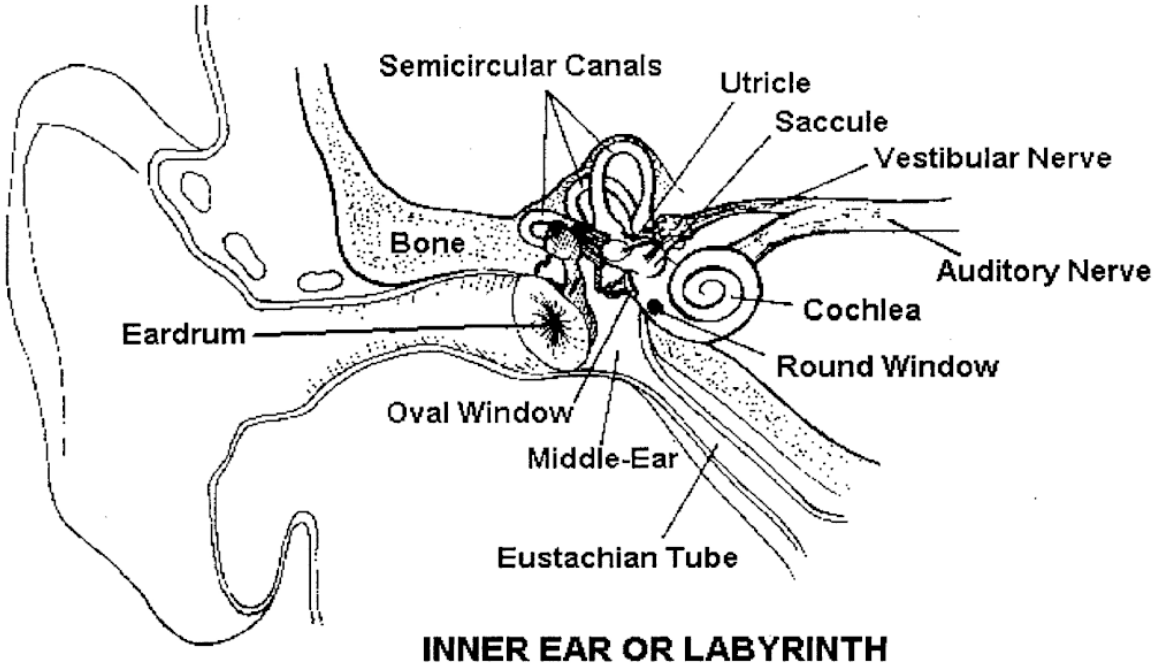



Figure 2.1: The components of the vestibular system [2].

Important for the sensory conflict theory are visual perception and the vestibular system, shown in Figure 2.1. The vestibular system consist of the Semicircular Canals to sense angular momentum, and the Utricle and Saccule to sense linear momentum. Together, the system functions to compensate for movement, stabilize vision, maintain head posture, and maintain balance [17].

In virtual environments, the sensory mismatch is usually between the visual system receiving optical flow patterns characteristic of self motion, while the vestibular system does not perceive these changes in motion. This sensory conflict lies at the root of simulator sickness and was identified early on, when Barrett and Thornton [18] noticed that subjects showed simulator sickness symptoms caused by conflict between the visual presentation of motion and the lack of corresponding vestibular sensation in their fixed-base simulators. Barrett and Thornton also noticed, that subjects only showed sickness symptoms when the simulator was in a perspective similar to driving a car, but showed no symptoms

when viewing the car from outside, similar to driving a remote controlled car, as well as passengers showing more severe symptoms than drivers, indicating that involvement in motion is a factor in the occurrence of simulator sickness [19, 18].

The sensory conflict theory is the most popular theory to explain cybersickness, because it has a steadily growing amount of studies supporting it, and the theory is intuitive to understand [1, 19]. However, the theory has been criticised by several studies, because sensory conflict theory only states that sickness is preceded by a sensory conflict, but is unable to predict when cybersickness will occur, or how severe sickness symptoms will be [2, 1, 20].

### 2.1.2 Postural instability theory

Another theory for cybersickness symptoms is the postural instability theory proposed by Riccio and Stoffregen [21]. They found that motion sickness is preceded by periods of postural instability, where small uncontrolled movements and changes in the subjects centre of gravity occur, and the subject's ability to maintain postural stability is hindered [21, 22].
Stoffregen and Smart [23] translated the theory into three predictions:

- Experiences of motion sickness are always preceded by increases in postural instability.
- Experiences of motion sickness persist until postural stability is restored.
- People who are more naturally unstable are more likely to become motion sick during provocative simulation.

These predictions have been solidified and are supported by numerous studies on visually induced motion sickness [22].

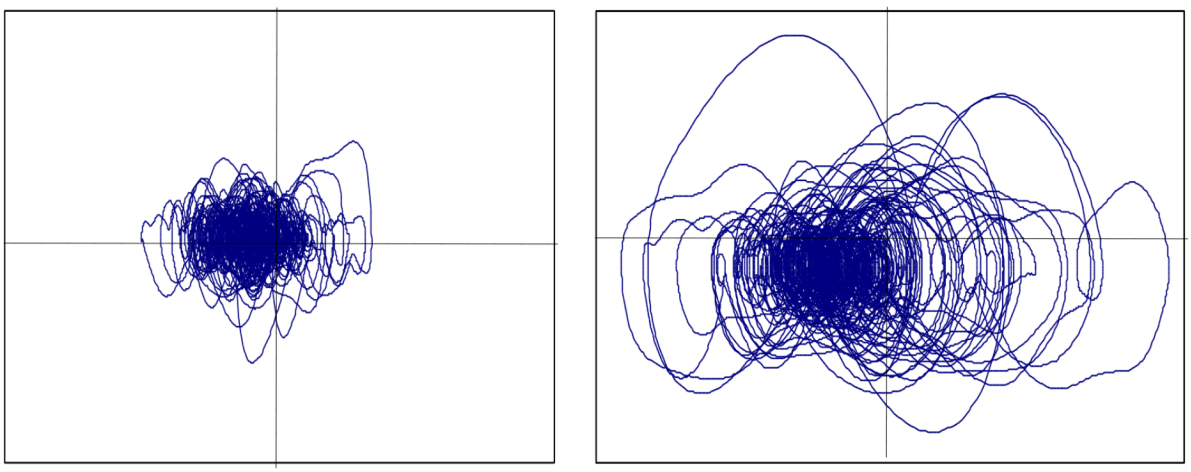


Figure 2.2: Comparison of phase portraits (position (in cm) vs. velocity (in cm/s)) for well (left) and sick (right) subjects in a dataset measuring postural stability [24].

Chardonnet, Mirzaei, and Merienne [25], as well as other studies propose to use the changes in range, variance, and frequency of the subject's centre of gravity as a measurement of postural sway.

Based on the accessibility of devices to measure individual's centre of gravity, those measurements have found increasing popularity in studies to objectively measure subject's postural stability and indicate the potential onset of cybersickness symptoms [26].
A comparison between the natural postural sway of a subject compared to the postural sway when experiencing motion sickness is shown in Figure 2.2.

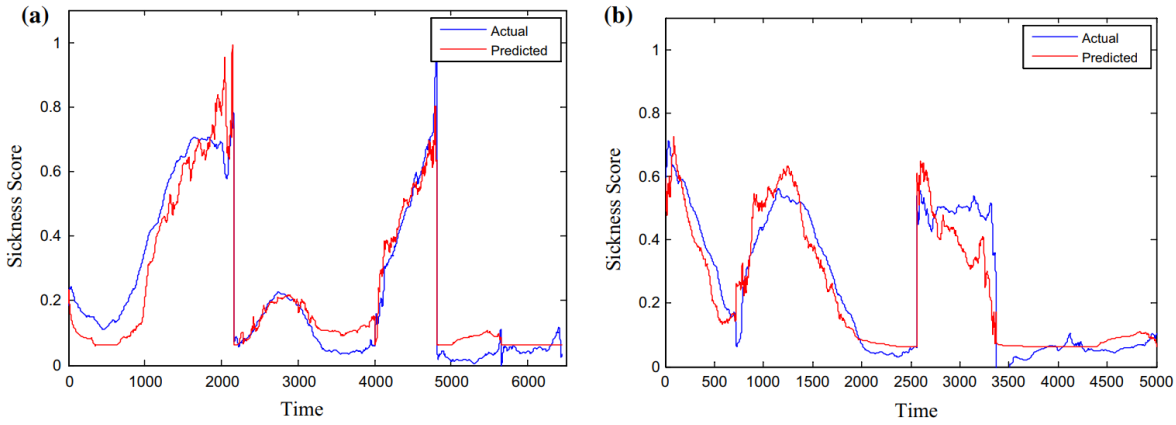



Figure 2.3: Actual sickness (blue) and predicted sickness (red) of (a) training and (b) testing set produced by the prediction algorithm by Lim et al. [26].

The recent study by Lim et al. [26] successfully used postural stability measurements to train an algorithm to predict VR content's potential to induce cybersickness, as shown in Figure 2.3, based on the postural instability theory.

### 2.1.3 Other theories

#### Rest frame theory

Similar to sensory conflict theory, the rest frame theory argues that a mismatch in sensed gravitation and perceived up-direction is the cause for sickness symptoms [1].

The rest frame theory also shows similarities to the postural instability theory, as the discrepancy between perceived up-direction and gravity leads to postural instability and following sickness symptoms [1]. An example of this sensory mismatch, and resulting postural instability is shown in Figure 2.4. The theory also supports the postural instability theory in situations where postural control is lessened, such as in seated positions where the individual's posture is stabilized. Several studies like Chang et al. [27], and Duh, Parker, and Furness [28] found, that superimposing some form of static frame of reference into the virtual environment significantly improves postural stability and reduces cybersickness symptoms.

#### Vergence-accommodation conflict theory

Another theory to explain cybersickness symptoms, especially oculomotor symptoms, is the vergence-accommodation conflict theory.

Vergence is the simultaneous lateral movement of the eyes when an individual's visual system is adjusting to objects at different distances [30]. Accommodation is the process of adjusting both eye's

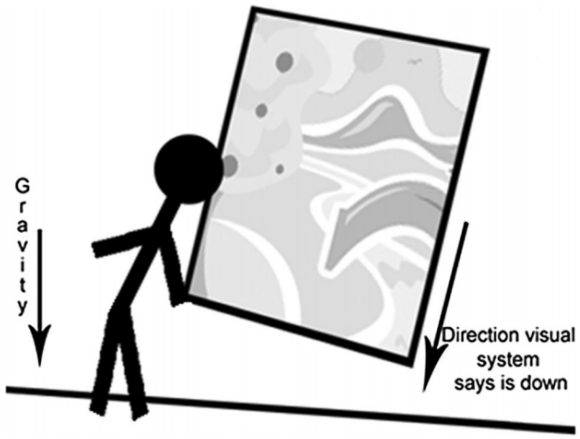



Figure 2.4: Example of sensory mismatch according to rest frame theory [1].

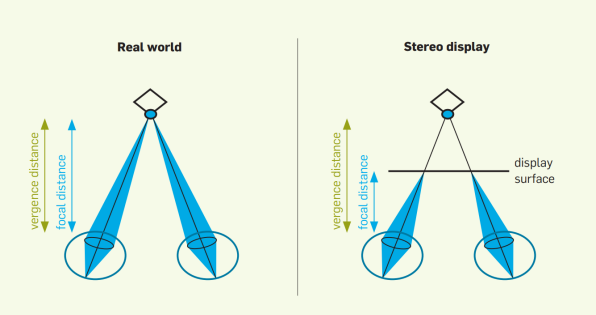



Figure 2.5: Difference between vergence and accomodation distance in the real world (left) and stereoscopic displays (right) [29].

focal lengths, focusing on the perceived object [1].
In virtual environments, especially in head-mounted displays, images are presented at a fixed screen depth. This leads to conflict with real life expectations, as vergence and accommodation do not occur naturally at different distances like in stereoscopic displays [8], as shown in Figure 2.5. Kim, Kane, and Banks [30] noted that content with high levels of stimulation usually contain more changes in stimulus distance, and therefore variance of stimulus distance, and level of visual stimulation both increase visual discomfort and eye strain.

## 2.2 Methods of measurement

Due to the polysymptomatic and polygenic nature of cybersickness, measurements of cybersickness can prove difficult, as there is a variety of mostly internal, nonobservable, and subjective symptoms [10]. Additionally, there can be large individual differences in symptom profiles and susceptibilty, and most symptoms develop over time and can occur even after the exposure to virtual environments [10].

Historically, the use of questionnaires is the most popular method of recording occurrences of

cybersickness [1, 8].
The most widely used questionnaire is the Simulator Sickness Questionnaire (SSQ), developed by Kennedy, Lane, Berbaum, and Lilienthal [9].
Recently, several studies have tried refining the SSQ to adapt it for the assessment of virtual reality and head-mounted displays, resulting in the Virtual Reality Symptom Questionnaire (VRSQ) by Ames, Wolffsohn, and McBrien [31], and the CyberSickness Questionnaire (CSQ) by Stone III [32].
Another popular questionnaire is the broader formulated Motion Sickness Assessment Questionnaire (MSAQ) by Gianaros, Muth, Mordkoff, Levine, and Stern [33], designed to assess visually-induced motion sickness symptoms regardless of stimulus context.
In addition to the extensive post-session questionnaires, single item questionnaires like the Fast Motion Sickness Scale (FMS) by Keshavarz and Hecht [34] that are polled in regular intervals during the virtual environment exposure have gained popularity in cybersickness studies.

Because of the subjective nature of questionnaires, these methods of quantifying cybersickness symptoms have been criticised and several methods of objective measurement have been researched. Kim et al. [35] studied the changes in sixteen different electrophysiological signals and found several measurements with a significant positive or negative correlation. However, these measurements require special equipment that may be unavailable or unintuitive, leading to a low adoption rate among studies related to cybersickness symptoms and detection.
A more easily accessible method of objective measurement has been measuring the postural stability of individuals exposed to virtual environments and use the changes in the centre of gravity (CoG) as an indicator for cybersickness symptoms [26].

### 2.2.1 Subjective Measurements

#### Simulator Sickness Questionnaire

The Simulator Sickness Questionnaire developed by Kennedy, Lane, Berbaum, and Lilienthal [9] is, despite being developed in 1993, still one of the most popular methods to measure cybersickness symptoms [8].
Kennedy et al. based their developments on the Pensacola Motion Sickness Questionnaire (MSQ), where they identified several deficiencies that could be improved:

- to provide a more valid index of overall simulator sickness severity as distinguished from motion sickness;
- to provide subscale scores that are more diagnostic of the locus of simulator sickness in a particular simulator for which overall severity was shown to be a problem;
- to provide a scoring approach to make monitoring and cumulative tracking relatively straightforward.

As part of the last objective, they sought to eliminate the configural approach of the MSQ allowing to automate the administration and scoring of results.
Additionally, the studies involving the MSQ used differences between post- and pre-exposure scores as their main indicator, and a pre-exposure checklist where subjects are asked whether they were in other than their "usual state of fitness". Kennedy et al. removed both, the two-step approach, and the

pre-exposure checklist in an effort to streamline the administration and scoring process, noting that the SSQ is "intended only for application to post-exposure symptoms, with the further precondition that a screening of "unhealthy" subjects is required [before exposure]" [9, p. 207].

To tailor the questionnaire to better fit simulator exposure and its sickness symptoms, Kennedy et al. eliminated symptoms that might give misleading indication, were selected too infrequently, or showed no change in frequency or severity.
Additionally, they sorted the remaining symptoms into separate clusters labeled "Oculomotor" (SSQ-O), "Disorientation" (SSQ-D), and "Nausea" (SSQ-N). The distinction allowed to apply a subscale to each cluster, reflecting the impact of simulator exposure on a different "target system" in the subject. It also simplified the process of determining where and in what way a simulator may cause problematic symptoms.

| | **Weight** | | |
|---|---|---|---|
| **SSQ Symptom** | *N* | *O* | *D* |
| General discomfort | 1 | 1 | |
| Fatigue | | 1 | |
| Headache | | 1 | |
| Eyestrain | | 1 | |
| Difficulty focusing | | 1 | 1 |
| Increased salivation | 1 | | |
| Sweating | 1 | | |
| Nausea | 1 | | 1 |
| Difficulty concentrating | 1 | 1 | |
| Fullness of head | | | 1 |
| Blurred vision | | 1 | 1 |
| Dizzy (eyes open) | | | 1 |
| Dizzy (eyes closed) | | | 1 |
| Vertigo | | | 1 |
| Stomach awareness | 1 | | |
| Burping | 1 | | |
| Total | [1] | [2] | [3] |
| Score | | | |
| $N = [1] \times 9.54$ | | | |
| $O = [2] \times 7.58$ | | | |
| $D = [3] \times 13.92$ | | | |
| $TS = [1] + [2] + [3] \times 3.74$ | | | |

Table 2.2: SSQ Symptoms and Scoring according to Kennedy et al. [9].

Table 2.2 shows the remaining symptoms and their clustering, as well as the method of scoring the SSQ as derived by Kennedy et al. The SSQ symptoms are rated on a four-point scale from 0 to 3, then multiplied by either 1 or 0 (omitted in the table) according to the weight section of table 2.2, and finally

summed up in each column. The total and subscale scores are then calculated using the formulas in the "Score" section of table 2.2.
Kennedy et al. also mention the possibility to further refine the questionnaire by:

- splitting the "Oculomotor" cluster into the disturbance of visual processing (blurred vision, difficulty focusing) and the symptoms caused by the disturbance (headache, eyestrain);
- splitting the "Nausea" cluster into premonitory signs (increased salivation, burping) and advanced stages of nausea (nausea, sweating);

As well as moving some symptoms into a "tired and hungry" cluster, believed to be an artifact created by the time spent during the exposure. However, Kennedy et al. state that there are not enough simulator-relevant symptoms to provide adequate reliability for smaller clusters, and recommend using the three cluster solution.
Despite the general adoption of the SSQ, there are several problems, especially for the assessment of cybersickness symptoms in virtual environments, that Kennedy et al. recognize in their study.

| | | | **SSQ** | **Scale M** | |
|---|---|---|---|---|---|
| **Simulator** | **Aircraft** | *N* | *O* | *D* | *TS** |
| 2F64C | SH-3 | 14.7 | 20.0 | 12.4 | 18.8 |
| 2F120 | CH-53E | 7.5 | 10.5 | 7.4 | 10.0 |
| 2F121 | CH-53D | 7.2 | 7.2 | 4.0 | 7.5 |
| 2F110 | E-2C | 7.1 | 13.1 | 6.8 | 10.3 |
| 2E7 | F/A-18 | 6.1 | 5.1 | 6.2 | 6.8 |
| 2F117 | CH-46E | 5.4 | 7.8 | 4.5 | 7.0 |
| 2F87F | P-3C | 4.5 | 15.2 | 4.3 | 10.5 |
| 2F132 | F/A-18 | 2.7 | 6.1 | 0.6 | 4.2 |
| 2F112 | F-14 | 1.7 | 1.8 | 0.0 | 1.5 |
| | | | | | |
| M | | 7.7 | 10.6 | 6.4 | 9.8 |
| SD | | 15.0 | 15.0 | 15.0 | 15.0 |

* Total Severity

Table 2.3: SSQ Scale Means by Simulator for the Calibration Sample according to Kennedy et al . [9].

The weights for the scoring functions are derived from 1119 pairs of MSQs collected from 9 simulator sites shown in table 2.3 as a calibration sample. Therefore, the modal position on the Symptoms or the intermediate sums is no indication for symptomatology with respect to simulator sickness across simulators in general, as the zero point contains between 40% and 75% of the observations in the calibration dataset. This also means, the sensitivity is at the upper extremes of the symptomatology range, and the scores should be compared to the calibration set, in table 2.3, instead of interpreted on their own [9].
Kennedy et al. conclude that the results should not be used to distinguish among simulators without problems, but identify and discriminate problem simulators from those without problems.
Other, more recent studies, like Sevinc and Berkman [36], and Rebenitsch and Owen [1], criticise the usage of the Simulator Sickness Questionnaire because of its complex structure, and development

process, as it involves only a sample of highly trained professionals, and a small amount of simulator experiments, which both do not comply with the modern day HMD-based virtual environments, and diverse applications and users [36].

#### Virtual Reality Symptom Questionnaire

Recent studies like Ames, Wolffsohn, and McBrien [31] tried to develop a questionnaire based on the MSQ and SSQ specifically for the assessment of cybersickness symptoms.
Ames et al. note that existing methods like the SSQ do not properly address the ocular symptoms that contribute to cybersickness symptoms in virtual environments. Examining existing virtual reality research, they identified 23 symptoms split into two clusters, 12 non-ocular, and 11 ocular symptoms.
Ames et al. also decided to expand the symptom response scale to seven options sorted into four labels: "none" (0), "slight" (1, 2), "moderate" (3, 4), and "severe" (5, 6).
In the development study, they exposed 16 subjects to a stereoscopic video played on a head-mounted display, and recorded the occurring symptoms with the developed VRSQ in two-minute intervals immediately after the exposure for a total of six post-exposure examinations. From the results, they identified 13 symptoms with high item-total correlations, that remain in the final questionnaire. While, the "nausea" symptom did not meet the correlation criteria, it was retained for research that might involve more "dynamic imagery".
However, similar studies, like Stone III [32], criticise the validity of the VRSQ to evaluate cybersickness, as Ames et al. only used video input on a head-mounded display, without any user interaction or input method, resulting in visual stimulus only, similar to existing studies on visually induced motion sickness, but not explicitly virtual reality sickness symptoms. Additionally, Stone III notes concerns about the validity of the psychometric evaluation, and the small sample size used in the development study.
Davis, Nesbitt, and Nalivaiko [37] also note the lack of published studies using the VRSQ as a method to evaluate cybersickness symptoms in their review on cybersickness literature, while Rebenitsch et al . [1] do not mention the VRSQ at all.

#### CyberSickness Questionnaire

In his study criticising the VRSQ and SSQ, Stone III [32] also proposes an alternative solution to measure cybersickness symptoms, the CyberSickness Questionnaire (CSQ).
Similar to the method Kennedy et al. [9] used to refine the SSQ from the MSQ, they reinterpreted the results of the SSQ in a cybersickness context. For this, Stone III selected the symptoms clearly indicating cybersickness:

- headache
- eyestrain
- nausea
- blurred vision
- dizzy (eyes open)
- dizzy (eyes closed)

- vertigo
- difficulty focusing
- fullness of head

Additionally, they decided to amalgamate "Severe" and "Moderate" responses from the SSQ.
Stone III found a two factor solution by separating the Symptoms into two clusters: "Dizziness" and "Difficulty focusing". They also note that the SSQ can still be used to record post-exposure symptoms, while the scoring can be done using the developed CSQ approach by following these steps:

1. Administer the SSQ after the exposure to virtual environments.
2. Remove the unneccesary symptom items from the collected data.
3. Combine "Moderate" and "Severe" options for each symptom item, resulting in responses "None" (0), "Slight" (1), and "Moderate" (2).
4. Compute the CSQ factors, similar to the SSQ, by multiplying each symptom item with the weight shown in table 2.4 and adding up the items to form the final scores.

| | **Dizziness** | **Difficulty focusing** |
|---|---|---|
| Headache | .50 | . |
| Eyestrain | . | .58 |
| Difficulty focusing | . | .89 |
| Nausea | .84 | . |
| Fullness of head | . | .55 |
| Blurred vision | . | .81 |
| Dizziness (eyes open) | .89 | . |
| Dizziness (eyes closed) | .99 | . |
| Vertigo | .54 | . |

Table 2.4: CSQ nine-item, two-factor model for scoring, according to Stone III [32]

Stone III notes that preliminary evidence, and the comparison of CSQ scores with other established visually-induced motion sickness scoring methods support the validity of the resulting CyberSickness Questionnaire. However, they note based on the CSQ scores of their study, 57% of the 202 participants reported no dizziness and 40% reported no difficulty focusing, which implies cybersickness was very low, and the study was not focused explicitly on inducing cybersickness symptoms.
The review of questionnaires by Sevinc et al. [36] has tested and approved the validity of the CSQ and concludes that it is a more accurate method of measuring cybersickness symptoms than the SSQ, as it was developed with a lager sample size and specifically based on the use of virtual reality applications to induce sickness symptoms.

#### Motion Sickness Assessment Questionnaire

Gianaros, Muth, Mordkoff, Levine, and Stern [33] developed the Motion Sickness Assessment Questionnaire (MSAQ) with the goal to assess motion sickness across a broad range of contexts.
Similar to other studies developing questionnaires, Gianaros et al. recognized the multi-dimensionality of motion sickness symptoms. However, they felt sopite-related symptoms, like drowsiness, yawning, and disengagement from the environment, were underrepresented in existing questionnaires [33, 38]. Including sopite-related symptoms, Gianaros et al. identified four dimensions of motion sickness:

- gastrointernal symptoms, like sickness, queasiness, or nausea;
- central symptoms, like dizziness, disorientation, lightheadedness, or blurred vision;
- peripheral symptoms, like being sweaty, clammy, or hot, or cold sweat;
- sopite symptoms, like being annoyed, tired, fatigued, or uneasy.

Similar to the SSQ, they designed the MSAQ to be used both, for overall motion sickness scores, and assessment of distinct dimensions via subscale scores.

| **Symptom Item** | **Gastrointestinal** | **Central** | **Peripheral** | **Sopite** |
|---|---|---|---|---|
| I felt sick to my stomach | .36 | . | . | . |
| I felt faint-like | . | .45 | . | . |
| I felt annoyed/irritated | . | . | . | .36 |
| I felt sweaty | . | . | .27 | . |
| I felt queasy | .36 | . | . | . |
| I felt lightheaded | . | .45 | . | . |
| I felt drowsy | . | . | . | .36 |
| I felt clammy/cold sweat | . | . | .27 | . |
| I felt disoriented | . | .45 | . | . |
| I felt tired/fatigued | . | . | . | .36 |
| I felt nauseated | .36 | . | . | . |
| I felt hot/warm | . | . | .27 | . |
| I felt dizzy | . | .45 | . | . |
| I felt like I was spinning | . | .45 | . | . |
| I felt as if I may vomit | .36 | . | . | . |
| I felt uneasy | . | . | . | .36 |

Table 2.5: Symptom items and subscale groups, according to Gianaros et al. [33]

The Questionnaire consists of between 16 and 20 symptom questions, with the original 16 questions shown in table 2.5 [33, 36]. Each symptom question is rated on a nine-point scale from 1 (Not at all) to 9 (Severly). The subscale scores are the sum of the weighted symptom items, multiplied by the weights in the respective columns, and the total motion sickness score is obtained by the following formula, according to Gianaros et al.:

- Overall Motion Sickness Score = ( sum of all symptom items (without weights) ) $\times$ 1.44

While, Gianaros et al. note that the generated symptoms during their study may stem from a relatively narrow range of motion sickness contexts, they offer the possibility to modify the questionnaire to more accurately reflect the multiple dimensions of motion sickness across different motion environments.
Although the development process of the questionnaire was based on visually-induced motion sickness using optokinetic drums and not virtual reality environments, Sevinc et al. [36] note in their review on cybersickness that the Motion Sickness Assessment Questionnaire was used in several virtual reality and simulator studies in the recent years. Lastly, Gianaros et al. argue that their questionnaire supplies valid descriptors of motion sickness in and for general population, since the symptom items were generated independently by non-experts during the early phases of their study.

#### Fast Motion Sickness Scale

In contrast to the multi-dimensional questionnaires, Keshavarz and Hecht [34] proposed and validated a single-item questionnaire that can be administered during stimulus presentation. This allows for simple and continuous gathering of motion sickness data during the presentation.
The FMS consists of a verbal rating every minute on a 20-point scale between 0 (no sickness) and 20 (frank sickness), and primarily measures the two cardinal symptoms in motion sickness: nausea, and general discomfort [34].
Keshavarz and Hecht explain, they use a 20-point scale to better differentiate among lower degrees of motion sickness symptoms and capture different states of both well-being and sickness. They argue the extended scale also helps participants to express their feelings and experience more precisely.
Conversely, Keshavarz and Hecht also note the questionnaire's indifference to the physiological correllates or root causes of motion sickness, and participants are unable to differentiate between nausea and other precursor symptoms in their answers. In their defense, they argue the Fast Motion Sickness Scale is only designed to quantify the subjective impressions of nausea and general discomfort related to motion sickness, without addressing the underlying symptoms or causes.

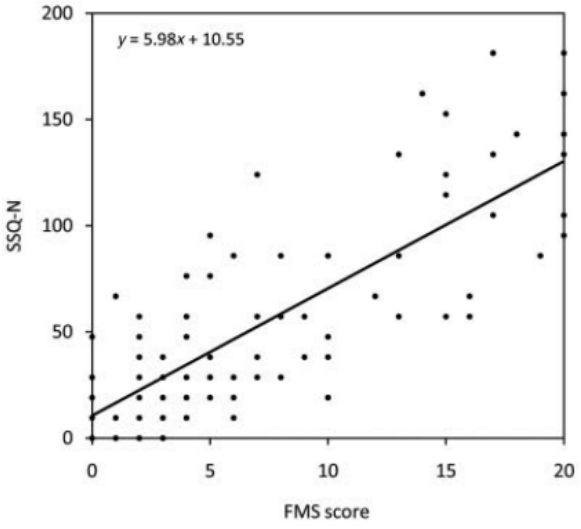



Figure 2.6: Scatter plot showing the distribution of peak FMS score and SSQ-N subscore for all participants of the study by Keshavarz and Hecht [34].

Their validation study showed high correlation between the peak scores of the FMS and the Nausea subscale score of the Simulator Sickness questionnaire (SSQ-N), as shown in figure 2.6.
Additionally, Rebenitsch and Owen [1] conclude that one-item rating scales are acceptable to monitor motion sickness symptoms, even though they are not as sensitive or thorough as the multi-dimensional, longer Questionnaires.
Sevinc and Berkman [36] also mention a rise in single-item assessment methods, but note that the FMS is the only version that has been psychometrically evaluated.

### 2.2.2 Objective Measurements

In contrast to the subjective measurements that have been employed in most studies on cybersickness, several studies have tried linking physiological measurements to the occurrence of cybersickness symptoms in individuals. Finding reliable physiological cybersickness indicators potentially allows monitoring symptoms without interrupting the virtual reality exposure through frequent polling of the subject, as used in the FMS, while still generating a continuous measurement stream to examine symptom development and causes over the duration of the virtual environment exposure [1].
One of the broadest studies regarding objective measurements of cybersickness symptoms is the study by Kim et al. [35], where they collected and tested 16 electrophysiological signals that were used in other studies as measurements of sickness symptoms in order to find significant correlations indicating a reliable, objective method measuring cybersickness.
The signals collected before, during, and after the exposure included the following items, which showed a significant correlation to reported sickness in previous studies:

- heart period
- respiratory sinus arrhythmia
- respiration rate
- eyeblink rate
- fingertip pulse volume
- fingertip temperature
- skin conductance
- gastric tachyarrhythmia (arrhythmic disruptions of the gastric musculature, often resulting in an upset stomach or uneasy feeling)
- electroencephalogram (EEG) power spectrum

Kim et al. used a quite provocative stimulus to test the physiological signals, as 45 out of the 57 subjects experienced cybersickness symptom during the 9.5 minutes of exposure to virtual environments and subjects reported cybersickness an average of five times during the exposure.
They also employed a 33-item pre-immersion, and a 49-item post immersion questionnaire to link to better link their electrophysiological findings to the self-reported levels of cybersickness symptoms experienced. The questionnaires were compilations of different popular cyber- and motion sickness and susceptibility questionnaires including the Simulator Sickness Questionnaire.

Kim et al. found significant correlations, especially between SSQ scores and gastric tachyarrhythmia, eyeblink rate, respiration rate, respiration sinus arrhythmia, and heart period. They found, gastric tachyarrhythmia, skin conductance, respiratory sinus arrhythmia, and relative delta power at F3 and T3 electrode locations of the EEG were significantly higher than the previous recorded baseline. They also found, heart period, fingertip skin temperature, fingertip pulse volume maximum amplitude, and relative beta power at F3 and T3 electrode locations were significantly decreased compared to the recorded baseline.
Kim et al. suggest that, due to the correlation between gastric tachyarrhythmia and cybersickness, the autonomic nervous system may play a bigger part in the occurrence of cybersickness than previously assumed. Similar to this study, Roberts and Gallimore [39] found electrogastrogram (EGG) measurements, to detect gastric tachyarrhythmia, to be a strong indicator of virtual reality exposure and cybersickness symptoms. Other studies have also used electrocardiogram (ECG) and blood pressure measurements to indicate cybersickness, as they found the heart beat becomes stronger, and the blood flow shows lower turbulence during exposure to virtual environments [40, 41].

Davis et al. [37] note that one detractor to widespread use of physiological measurements may be the costly hardware and difficulty to analyse the results. A middle ground has been presented in several studies using measurements of postural stability as an objective, low cost, and continuous indicator of cybersickness symptoms [1].
However, Rebenitsch et al. [1] also note that postural sway measurements in many studies are not as continuous or without the disturbance of the subject as they are often presented, since most measurements require the subject to enter a specific stance for the measurement.
Most studies use the amplitude, magnitude, and frequency of postural sway to distinguish and predict motion sickness symptoms, while older studies also examine the time till failure (time until a stance cannot be maintained) or the stance breaks (number how often the subject fails to maintain a stance) during a specific timeframe [1].
Villard, Flanagan, Albanese, and Stoffregen [42] and Dong, Stoffregen, and Yoshida [43, 44] both noticed in their studies that subjects who did not experience significant sickness symptoms had an increased standard deviation of motion during their exposure period, while subjects that experienced sickness symptoms showed greater variability with smaller average motion compared to the baseline. Other studies mentioned similar findings and suggest that subjects may self-adapt to virtual reality exposure by consciously avoiding head movements when experiencing sickness symptoms [1].

Another method of low cost, objective measurement of cybersickness is presented by Chardonnet et al. [25], who found consistent results measuring a subject's centre of gravity (CoG) with a high correlation to measured SSQ scores. Lim, Lee, Won, Kala, and Lee [26] further extended this measurement, as they proposed to use the VR device's inertial measurement unit (IMU) to record head dispersion during the exposure period.
They introduced Head Dispersion as a measurement of stability from the following Eq. 2.1

$$Head\ Dispersion = \sqrt{\frac{\sum(roll - \overline{roll})^2 + \sum(pitch - \overline{pitch})^2}{n}} \tag{2.1}$$

With the $roll$ and $pitch$ values in degree, and $\overline{roll}$ and $\overline{pitch}$ as the mean values along the session.
To validate their approach Lim et al. compared their IMU sensor data to the centre of gravity sway area

measured by an external sensor as proposed by Chardonnet et al. and found high correlation between their measured Head Dispersion and CoG sway area.
The major advantages of this method are the lack of additional measuring devices needed, and the synchronization processing, while delivering a continuous indicator of potential sickness symptoms. However, due to the study's recency, it has not yet seen further adoption or larger sample size studies.

## 2.3 Methods of Mitigation

Despite the lack of a definitive cause or widespread adoption of objective measurements to monitor cybersickness symptoms, several mitigation techniques and best practices have proven effective and have found widespread adpotion as early as 1992, when McCauley and Sharkey [10] formulated their best practices and recommendations to prevent and mitigate cybersickness and simulator sickness symptoms:

*Exposure time should be limited until adaption to the VE has occurred*, as some studies found that users adapt to virtual environments with repeated exposure [7].

*Tasks that require high rates of linear or rotational acceleration should be avoided, or kept brief, until the individual has fully adapted to the altered environment*, as virtual reality content with high interaction and visual stimulation tends to lead to more severe experiences of motion sickness [8].

*Users of VEs should be considered on an individual basis when determining an adaptation program*. While this recommendation is focused on adaption programs for simulators, it is also applicable to virtual reality users, as they show individual differences in cybersickness susceptibility or, for example, preferred locomotion method [22].

*Self-movement through a VE should be at high altitudes above the terrain and/or at lower speeds*, as high peripharal motion and visual flow are related to cybersickness occurrence and severity [45].

Additionally, *unusual and extraordinary maneuvers should be avoided in VEs*, as abrupt and counterintuitive movements can further disorient the user, increasing the risks of cybersickness.

Finally, *users of VE systems should be informed of the possible adverse effects and should be advised to allow for recovery time after cybertravel before actively engaging in potentially dangerous activities in the real world, such as driving*, since studies have shown that cybersickness symptoms can occur delayed after the exposure and have potentially lingering effects [2].

Apart from these general recommendations, several techniques have found widespread adoption and have shown a positive impact in mitigating cybersickness symptoms, like the limitation of the Field of View, or the insertion of stable frames of reference into the virtual scene.

### 2.3.1 Field of View Limitation

The interaction between Field of View (FoV), cybersickness, and the individual's feeling of presence, have been the subject of many studies, trying to identify the connections and possibly find an optimal

solution for the FoV, where cybersickness symptoms are minimised, while maintaining the feeling of presence as much as possible [46].
Duh et al. [47] found vection to be strongly tied to FoV size, as subjects mainly seem to receive information about vection from their peripheral visual field. Similarly, Lin, Duh, Parker, Abi-Rached, and Furness [48] examined the connections between FoV and several other key aspects of virtual environments, including presence and cybersickness, where they found subjects report increasing SSQ scores and feeling of presence with increasing FoV. Therefore, a wide FoV causes a greater perception of self-motion, which tends to lead to increased postural disturbance, and generally resulting in more severe cybersickness symptoms.

Studies like Fernandes and Feiner [49] recommend to dynamically, or at least strategically, manipulate the Field of View in order to reduce visual flow in the peripheral field during periods of high visual flow, reducing the impact of the resulting vection on cybersickness symptoms.
Duh et al. examined the limits of narrow Field of Views and found that, while the experienced motion sickness increased with higher FoV, there is a significant increase at the 120°-150°interval.
Additionally, participants in the study by Lim et al. [26] reported "less noticeable" black regions until the FoV dropped below 60°. Lim et al. conclude that FoV limitations should be individually adjusted, as users react differently to varying degrees of FoV limitations, as well as dynamic and fixed FoV limitation.
Finally, they note that, according to their study, dynamic FoV processing and limiting can decrease VR sickness symptoms by up to 37%.

### 2.3.2 Stable Reference Frame

Another common countermeasure for cybersickness is based on the rest frame theory, which also includes some aspects and links to both, postural instability and sensory conflict theory. The measure is focused on providing the user with a stable frame of reference in the virtual environment, in order to enable the user to find the correct up-direction and therefore minimising sensory conflicts and improving postural stability.

Several studies, like Duh et al. [28] and Chang et al. [27], superimposed grid lines into the virtual environment and found significant improvements in cybersickness symptom ratings.
Kemeny, George, Mérienne, and Colombet [50] note in their study that displaying a fixed reference frame helps users to better understand and process rotations as it creates a "Pseudo AR Mode", where reduced sickness symptoms are traded for a lesser feeling of presence.
A similar study by Kato and Kitazaki [51] employed a superimposed grid on in-vehicle displays to mitigate the sensory conflict in a moving car and delay the onset and reduce the severity of carsickness.
Finally, Clifton et al. [22] also recommend the provision of a stable simulated reference frame to reduce cybersickness especially during virtual travel and for unstable individuals.

# 3 Current State and Problems of CosmoScout VR

In this chapter, special concepts and features of the CosmoScout VR application are presented, which shape and influence the methods and solutions to mitigate cybersickness symptoms presented in chapter 4.
Additionally, situations and aspects of CosmoScout VR that have shown to provoke cybersickness in the simulation are examined in the second section of this chapter to highlight the areas that the implemented solution are targeted at, in order to alleviate the problems and provide a more comfortable user experience without unnecessarily limiting the user.

## 3.1 CosmoScout Concepts

### 3.1.1 Control Scheme

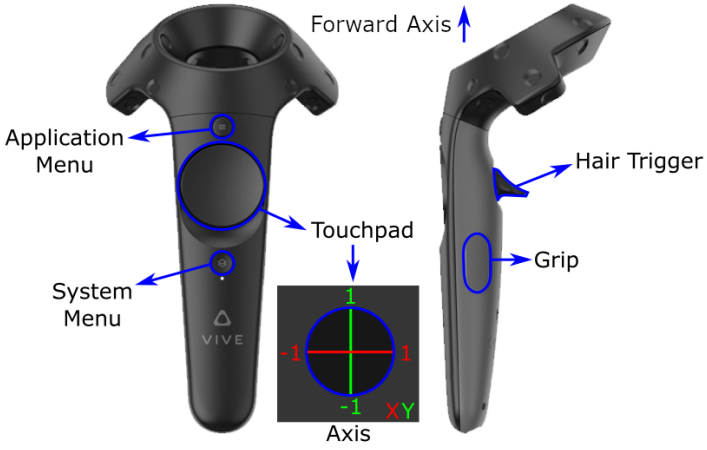



Figure 3.1: HTC Vive Remote Controller Button reference [52].

The main control scheme of CosmoScout VR for HMDs employs a one-handed flying scheme similar to the control scheme of Google Earth VR or the One-Hand Flying described by Drogemuller et al. [53]. The controls use the buttons highlighted in figure 3.1, as well as an indicator that is drawn as an extension of the remote controller's forward pointing axis.
Pressing the touchpad allows the user to rotate the simulation around the body the user is pointing at with the remote's indicator, i.e. the observer is moved around the body in a circular orbit. The angle of the rotation corresponds directly to the magnitude of the vector between the initial and current position of the indicator's intersection with the body.
Pressing the grip button allows the user to "grab" the body, the remote indicator is pointing at, and move the body on a circular orbit around the observers position. The observer is always rotated so that

the "grabbed" point on the body remains at the position of the remote indicator. Pulling the hair trigger allows the user to move the observer freely with 6 degrees of freedom (see figure 3.2).
While the trigger is pulled, the observer's translational movement is determined by the vector from the initial position when the trigger was pulled to the current position. The vector's magnitude directly corresponds to the velocity of the observer in the direction of the vector. Additionally, while the trigger is pulled, the observer's rotational movement is determined by the angles between the initial and the current orientation of the remote. The angles correspond to the angular velocity of the observer's rotation. Through the translation of vector magnitude and angles of orientation into velocities, the user does not need to twist the remote in uncomfortable positions, but can gradually navigate the simulation with small, precise movements.

### 3.1.2 SPICE Coordinate Systems

## 3.2 Identified Problems

In order to improve usability and reduce cybersickness symptoms, we aimed to identify and improve aspects and scenarios in CosmoScout VR that led to cybersickness symptoms or general feeling of discomfort.
To identify these problems, we conducted small tests, gathering empirical data, and used small interviews and discussions with other developers and users of CosmoScout. A formal study with unbiased test subjects was not done for several reasons, mainly since the goal is only vaguely defined and may require a decently sized sample size, as well as time to process its results. Additionally, the health and safety regulations and lockdowns around the COVID-19 outbreak complicated the feasibility of any preliminary study.
The empirical data and interviews identified two major problem areas that need independent solutions to mitigate discomfort and sickness symptoms. The 6-Degrees-of-Freedom (6-DOF) in interplanetary space can easily lead to visually induced motion sickness, and the rotations and translations of automatic navigation similarly have a tendency to lead to motion sickness symptoms, especially the descending and ascending animations to navigate to a body's surface.

### 3.2.1 Problems with free movement

While traditional 4-DOF-navigation provides an inherently more stable frame of reference with the fixed plane of movement usually providing an up-direction strongly tied to the movement scheme, a 6-DOF-navigational system lacks this reference frame as it allows the user to freely rotate around all axes. A reference to the different degrees of freedom is shown in figure 3.2.

Keshavarz, and Hecht [55] examined the influence of rotations around multiple axes and found that increasingly complex rotations lead to an increase in cybersickness symptoms. Additionally, Rebenitsch et al. [1] found multiple studies examining the effects of different degrees of freedom in navigation. They mention 6-DOF navigation usually induce more cybersickness symptoms than navigation with limited degrees of freedom and conclude that the reason might be a limitation of rotation axes.
The control scheme of CosmoScout VR as described in section 3.1.1 does not allow for easy and independent control of rotation around each axis, which can quickly lead to complex rotations around multiple axes at once. These complex rotations paired with a lack of reference frame can easily disorient the user and lead to visually induced motion sickness symptoms.

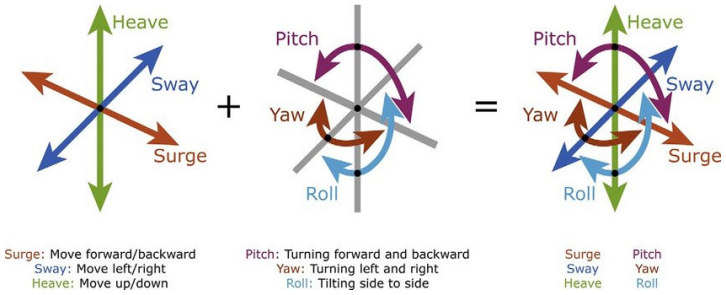



Figure 3.2: Six potential degrees of freedom for rotation and translation of an object [54].

While it seems both problems originate from the control scheme, a change of the control scheme does not necessarily help mitigate the problems as different control schemes like those proposed by Drogemuller et al. [53] either suffer from the same problems, or significantly change the interaction context. Isolating the rotation controls would be hard to implement with only the VR remotes and lead to a more unintuitive and inconvenient control scheme. Therefore, other mitigation methods are sought to provide the user with a stable frame of reference and reduce the impact of complex rotations.

### 3.2.2 Problems with automatic movement

The current automatic navigation to selected bodies or bookmarked locations was only designed as a rudimentary method to facilitate the means of quick and precise movements to points of interest without the need for manual travel to said point of interest.


Figure 3.3: Current automatic movement between two points of interest.

The navigation uses linear interpolation between the current position and rotation, and the target position and rotation to change both variables over a predetermined duration. Additionally, the animation speed is variable to start the movement slow, speeding up in the intermediate travel, before slowing down again, settling into the final position and orientation. Figure 3.3 describes the movement process between two points of interest during the automatic navigation. The color gradient represents the velocity of the observer in the simulation during their linear movement. The camera icons represent the rotation of the observers field of view, and the frequency of the icons roughly represent the angular velocity of the rotation.
Obviously, a complex rotation paired with linear movement can easily provoke visually induced motion sickness in users experiencing these movements, as these motions resemble provocative simulations used in several studies to actively induce cybersickness in their subjects.

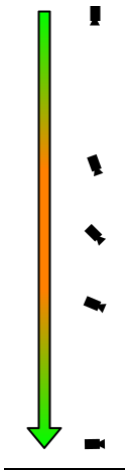


Figure 3.4: Current automatic movement descending onto a body's surface.

The movements to ascend or descend between a body's orbit and its surface have been mentioned by some users to be a significant problem of the automatic movement in CosmoScout VR and source of discomfort. The source of increased cybersickness symptoms is thought to be the more prominent reference frame provided by the body's surface.
As presented in figure 3.4, the movements use the same interpolation as mentioned above. However, during the intermediate period, where both linear velocity of translation and angular velocity of rotation are highest, the perceived reference frame changes.
While the observer is positioned in a body's orbit, the user's frame of reference is oriented such that the body is "in front" of them, i.e. the surface's normal being perpendicular to the perceived up-direction. During the descending movement the body's surface normal shifts to be parallel to the up-direction without the user vestibular system feeling angular acceleration or a shift in gravity. Paired with the disorientation during the automatic movement this strong reference frame and change of orientation may exasperate the resulting motion sickness symptoms, as the rotations in interplanetary space only have other bodies, and the surrounding stars as weak points of reference to indicate the rotation.
The ascending movement suffers from the same problem, but in reverse, changing the body's surface from a reference plane parallel to the real world surface to it being perpendicular to perceived gravity.

Finally, the current navigation does not have any collision detection or handling, which can contribute to discomfort of users when the linear path of the navigation intersects a body, as shown in figure 3.5.

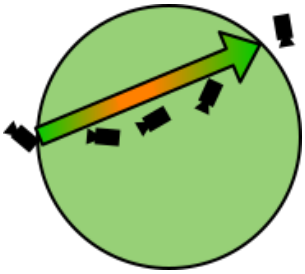


Figure 3.5: Current automatic movement between two points of interest on different sides of a body.

While, collision detection and handling is not part of this study, we aimed to mitigate collision related problems in our solutions and provide navigation methods that allows collision detection and handling to be added afterwards without major changes to the basic structure.

# 4 Implemented Solutions

## 4.1 Floor Grid

As suggested in the studies examined in section 2.3.2, we decided to add a stable reference frame to help with postural stability. Especially, we focus on the 6-Degrees-of-Freedom navigation in the interplanetary areas of the Simulation, that generally lack any sort of fixed reference frame, since movement on a planet surface has the surface and its normal direction as a strong visual indicator for a reference frame.
The floor grid aims to provide a definitive up-direction that is parallel to the real world's gravity, thereby improving postural stability and minimizing sensory conflicts.
While superimposing a grid on the floor may significantly reduce the feeling of presence, we accept this effect since the primary function of CosmoScout VR as a scientific visualization tool is not as dependent on an immersive feeling as entertainment oriented simulations may be.
Additionally, we also suspect that small changes to the control scheme can be done to further change the interaction context from an egocentric view within the simulation to a more exocentric "Pseudo AR" approach, where the user manipulates the surrounding simulation without moving, similar to the Worlds-in-Miniature approach proposed by Drogemuller et al. [53].
The Hypothesis is that a "Pseudo AR" simulation environment would drastically reduce cybersickness symptoms, as the sensory conflict between the user inside the simulation and the real world is minimal, and maintaining postural stability should be easier for the user.
Additionally, the best practices mentioned by McCauley et al. [10], include that users should be considered individually as reaction and adaptation to virtual environments are different for each case. In order to match individual users adaptation progress, as well as different scenarios regarding the hardware setup, most values and aspects of the grid can be customised through the settings file and almost all of them are also changeable in the options menu at runtime.

### 4.1.1 Floor Grid Implementation

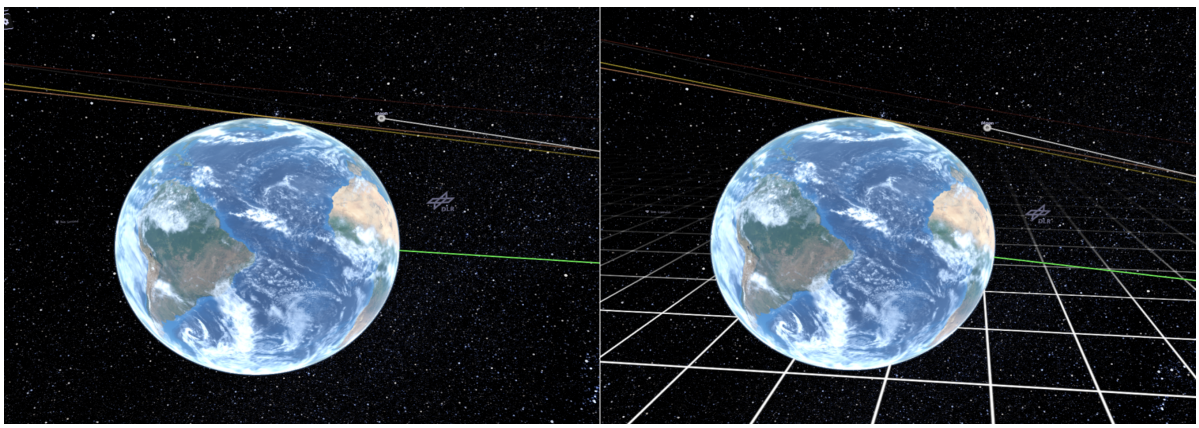


Figure 4.1: Screenshot of CosmoScout VR without (left) and with (right) the Floor Grid.

The Floor Grid is implemented in the VR Accessibility Plugin for CosmoScout VR, so it can be optionally loaded with CosmoScout when it is needed and can be excluded in hardware setup where they are not.

It renders a quad that is fixed to the observer's position and GUI elements. This way still allows free movements with the VR HMD while staying a static frame of reference that is tied to the real world floor.
To properly adjust the grid to the real world floor the node drawing the quad is attached to the node drawing the GUI via a transform node in the scenegraph that adds a vertical offset (in meters) to the grid that is adjustable in the settings file before startup, and the options menu at runtime.
The quad size is defined in the settings file as a multiplier (*uFalloff*) that scales the initial quad, spanning in x and y direction from -1 to 1, up by the value set in the settings file.
The scaled quad is textured with a semi transparent texture containing the grid with the center of the grid either as a square or a cross. Additionally, the texture can be scaled by a separate factor (*uSize*) resulting in a coarser or finer mesh.
The calculations for the vertex positions and texture coordinates are shown in the code block below from the Floor Grid's vertex shader:

```
1  void main()
2  {
3      vTexCoords  = vec2( (iQuadPos.x + 1)/2 * uFalloff * uSize,
4                          (iQuadPos.y + 1)/2 * uFalloff * uSize );
5
6      vPosition   = (uMatModelView * vec4(iQuadPos.x * uFalloff, 0.0, iQuadPos.y * uFalloff,
       ↪  1.0)).xyz;
7      gl_Position = uMatProjection * vec4(vPosition, 1);
8  }
```

The variables *uMatModelView* and *uMatProjection* are the Model, View, and Projection Matrices needed to transform the coordinates in the graphics pipeline.
The Floor Grid's fragment shader applies the selected texture, but is discarded if the fragment is not part of a grid line (i.e. is fully transparent). Additionally, the texture's opacity (*uAlpha*) is adjustable through the settings file and in the options menu at runtime.
To prevent aliasing or the grid turning into a grey plane towards the horizon, the floor grid is drawn with the OpenGL Blend Function "`glBlendFunc(GL_SCR_ALPHA, GL_ONE_MINUS_SRC_ALPHA)`". This blend function leads to the grid fading after a few meters because the texture is mostly transparent. finally the color of the grid is adjustable in the settings file or in the options at runtime to allow more customisation for individual users.
Figure 4.1 shows a screenshot of the floor grid in use compared to the same scene without the grid.

## 4.2 Field of View Vignette

One of the most common methods to reduce cybersickness risks and symptoms is decreasing the Field of View as examined in section 2.3.1.
To alleviate cybersickness during movements with high detail, and movements in the peripheral areas of vision, a vignette is implemented to limit the Field of View, focusing the users attention and preventing the influence of activity in the peripheral vision from adding to cybersickness symptoms.
The Vignette is primarily planned for movements close to an object's surface, where peripheral visual flow is significantly higher compared to movements in interplanetary space. Additionally, reducing the FoV may help users during the adaption phase, by directing their view to the center of the HMD's lenses where the image is focused.
While a point of interest is within a users field of view, they tend to shift their gaze without moving their head. This can lead to users trying to focus on objects that are not at the center of their view, and thereby not in the center of the HMD lenses. In this case the user is trying to focus on part of the virtual environment, that cannot be focused on based on hardware limitations. A FoV vignette can help the adaptation process by limiting the FoV and nudging the user towards using head movements instead of eye movements to look at different points of interest and keep their gaze centered at the middle of the HMD's lenses.
Drogemuller et al. [53] also recommend dynamically changing the FoV while using a one-handed control scheme to reduce a users feeling of presence during complex movements in order to reduce discomfort. However, these adaptation aids require a narrow FoV and may be perceived as limiting or annoying to other users.
According to the best practices by McCauley and Sharkey [10], and similar to the Floor Grid, the vignette is designed to be modular and adaptable to each individual user. Additionally, several options of vignetting are implemented, as suggested by Lim et al. [26], to allow the user options for both dynamic and fixed limitation of FoV.

### 4.2.1 FoV Vignette Implementation

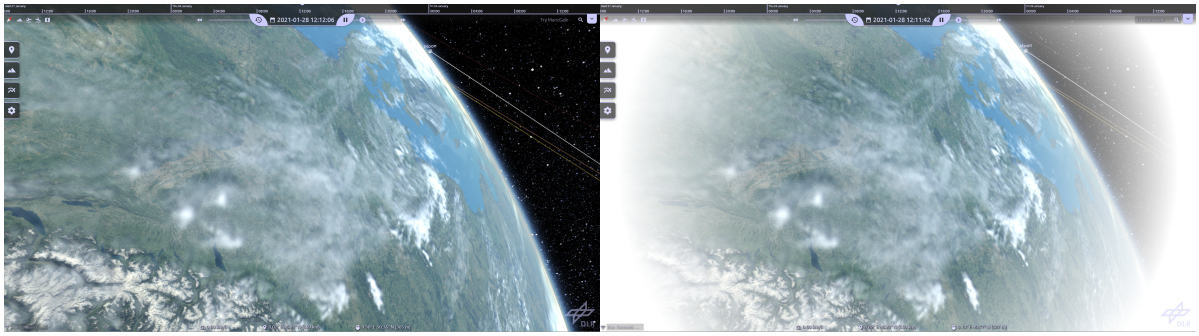


Figure 4.2: Screenshot of CosmoScout VR without (left) and with (right) the FoV Vignette.

The FoV Vignette is also implemented in the VR Accessibility Plugin for CosmoScout VR, together with the Floor Grid.
The vignette is realized as a post-processing shader, drawing a 2D effect over the rendered scene based on an inner and outer radius, which are both adjustable in the settings.

The inner radius determines the maximum distance from the center of the viewport, where a clear field of view is guaranteed. The outer radius determines the minimum distance from the center, above which the screen is fully opaque with a custom color that is also adjustable in the settings. The area between the inner and outer radius is bridged by a gradient, blending between fully transparent, showing the rendered scene, and the fully opaque custom color at the outer radius.
Since the vignette is designed to block peripheral visual flow inducing vection and distracting the user during movement, the vignette is only drawn during movement, and disabled when standing still or during sporadic, short movements. This allows reducing the risk of cybersickness symptoms during critical phases, while still maintaining the feeling of presence as much as possible.
As mentioned above, two different methods of vignetting are implemented to offer more customization to the user, a dynamic and a static vignette.

One method is drawing the vignette dynamically based on the observer's velocity in relation to the Observer's position and distance to other bodies, as interplanetary movements allow for, and use significantly higher velocities than movements close to a bodies surface.
For more customisation, an adjustable upper and lower velocity threshold for this normalized velocity is given to allow more sensitive users an earlier and more noticeable vignetting, and more robust users the freedom to reduce its effects. The lower velocity threshold is the velocity at which the vignette will start limiting the FoV by decreasing its radius, and the upper velocity threshold is the velocity where the vignette is at a determined minimum radius. An adjustable threshold for the minimum velocity is used, since slow movements tend to only produce low risks of cybersickness symptoms and limiting the FoV is usually not necessary. Providing an upper velocity threshold serves solves two problems, it allows sensitive users to benefit of a drastically reduced FoV as early as they feel comfortable, and additionally preventing the vignette from jittering at maximum velocity where the velocity tends to fluctuate slightly, propagating to the radii of the dynamic vignette.
The FoV vignette is drawn in the same way for both options as shown in the fragment shader for the dynamic radius:

```
1  void main()
2  {
3      if (uNormVelocity <= 0 && !uDebug ) { discard; }
4
5      vec2 texSize = textureSize(uTexture, 0);
6      float ratio = texSize.y / texSize.x;
7
8      float dist = sqrt(vPosition.x * vPosition.x + ratio * ratio * vPosition.y * vPosition.y);
9
10     if (dist < uInnerRadius ) { discard; }
11     float r = ((dist - uInnerRadius) / (uOuterRadius - uInnerRadius));
12     oColor = mix(texture(uTexture, vTexCoords), uCustomColor, r);
13     if (dist > uOuterRadius) {
14       oColor.rgb = uCustomColor.rgb;
15     }
16 }
```

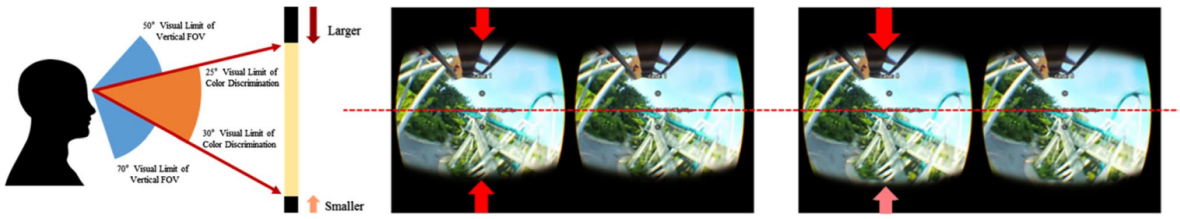
The debug flag (*uDebug*) exists to allow for easy adjustments of the vignette's variables, and if enabled draws the vignette with its minimum radius.
First the distance (*dist*) of the current fragment to the center of the viewport is calculated and adjusted for the viewport's aspect ratio (*ratio*). Based on this distance and the inner and outer radius (*uInnerRadius*

and *uOuterRadius*) the blending factor ($r$) is calculated, and the rendered scene (*uTexture*) is blended with the custom color to create the gradient between the inner and outer radius. If the fragment is inside the inner radius, the post processing is discarded,a nd if the fragment is outside the outer radius it is set to the color of the vignette (*uCustomColor*).
The dynamic vignette's radius is controlled through the inner and outer radius that is passed to the fragment shader, and is updated during the update frames of the simulation. In the update the velocity is normalized to the upper and lower velocity threshold, and a new radius is calculated partly based on the previous radius to dampen the propagation of harsh changes in velocity and smoothen the changes in radii. Currently, new radii propagate with a 1% influence over each frame:

$$r_{new} = (0.99 \times r_{last}) + (0.01 \times r_{target}) \tag{4.1}$$

As suggested by Lim et al. [26], a second option with a static vignette is also implemented.
The static vignette also uses the lower velocity threshold to determine when it should be drawn.
Additionally, an adjustable deadzone is implemented, allowing for a grace period where the vignette is not displayed when passing the threshold to avoid flickering on short, quick movements, or velocities close to the threshold, that pass the threshold when fluctuating slightly.
After passing the velocity threshold for at least the time specified in the deadzone or longer, a fade animation is used to ease the vignette in or out over an adjustable duration, to make the transition to the limited field of view more comfortable and less noticeable.
Through the use of an animation to gradually increase the opacity of the Vignette, a closing movement similar to the dynamic vignette along the gradient is suggested. However, the vignette is independent of the velocity, and is drawn with its specified minimum radius as soon as the lower velocity threshold is passed.
As with the Floor Grid, almost all settings specifying the FoV vignette are adjustable either in the settings file, or in the options menu at runtime, to allow users to customise the vignette according to their hardware setup and preferences.

Finally, an experimental option is implemented, to only limit the FoV vertically and asymmetrically.




**(a)** Symmetric FOV **(b)** Asymmetric FOV

Figure 4.3: Asymmetrical FoV Limitation as proposed by Lim et al. [26].

Lim et al. [26] proposed in their study that the FoV vignetting should be applied asymmetrically, as human visual characteristics produce an asymmetrical field of view with visual limits of 50°upwards and 70°downwards [56].

Since the horizontal field of view is significantly wider than the vertical field, we added an option to asymmetrically limit the vertical field of view and forego any limitation of the horizontal visual field.
The fragment shader for the vertical vignette functions the same as their respective versions for the round vignettes. However, the calculation of the distance from the fragment to the center ignores the horizontal x-component as depicted below:

```
float dist = 0;
if (vPosition.y > 0) { dist = vPosition.y; }
else { dist = vPosition.y * -(0.7); }
```

instead of calculating the euclidean distance, the distance above the center is the vertical position of the fragment from the center. The distance below the center is multiplied by 0.7 to result in a smaller distance, and thus a smaller vignette on the bottom, as depicted in figure 4.3.
This feature is implemented experimentally, mainly because Lim et al. implemented an asymmetrical FoV Limitation in their study, but no symmetrical limitation.
We understand that Lim et al. as well as their reference Hatada, Sakata, and Kusaka [56] argue for an asymmetrical FoV vignette as it is based on human visual characteristics, and is therefore less noticeable. However, an asymmetrical vignette could lead to users not focusing on the center of the HMD's lenses, but slightly below, leading to unfocused images that may increase cybersickness symptoms.

## 4.3 Automatic Movement Overhaul

As pointed out in section 3.2.2, the current automatic movement method was built as temporary means to navigate between saved bookmarks. The method applies a linear interpolation to transition smoothly between the origin and target position leading to movement in a straight line between the two points. Additionally, a linear interpolation between the origin and target orientation is used to rotate the observer into the target orientation.
While this is the easiest, and most straightforward way to smoothly move the observer through the simulation without breaking continuity by teleporting the user, the simultaneous translation and rotation have been found to induce disorientation followed up by motion sickness symptoms by the user.

In this section a new automatic navigation for CosmoScout is presented, which aims to make the automatic navigation more accessible and less sickness inducing. As parts of the navigation have not been fully implemented, we first present the concepts how the automatic navigation system should move the observer. After the concept, key features of the implemented solutions are presented and explained.

### 4.3.1 Automatic Movement Concept

The main goal, found to reduce cybersickness for automatic movement, is reducing the complexity of the movement by decoupling the rotation from the translation. This way the movements are easier to read, and the user should be able to better anticipate the observers movements, reducing motion sickness symptoms.
First, the linear interpolation is exchanged for movement along a spline. This way, it does not change the linear movement between points in space as it was implemented before, while still accomodating for different, non-linear paths the observer can take. Additionally, this change is made to facilitate more variability for later development, like programmed virtual tours along points of interest. Additionally, the change to splines is made to enable easier collision handling for the navigation method, as additional control points can be inserted to handle collisions and divert the movement path.

Since the origin and target location and orientation of the automatic navigation can be arbitrary points anywhere in the simulation (both bookmarked and calculated intermediate locations), the first step is to classify the possible locations into groups. We propose the classification of locations based on their location relative to other bodies, and the SPICE reference frame the location is in:

- Surface locations, that are close to, or on the surface of a body, where the location's position and orientation are in the respective body's SPICE frame relative to the body's center;
- Orbit locations, where the location is not on or close to the surface, but the position and orientation are still relative to the body's SPICE frame and center;
- Interplanetary locations, where the location's position and orientation is not relative to another body's SPICE frame and center, but the solar system's barycenter, i.e. the "J2000" SPICE frame.

Through the classification of locations a set of general movements and transitions can be derived to access all of those locations in a manner that is predictable for the user, while still general enough to minimize special cases that could lead to unwanted behaviour.

The classification leads to 3 types of movement and the transitions between them, resulting in the state machine shown in figure 4.4.

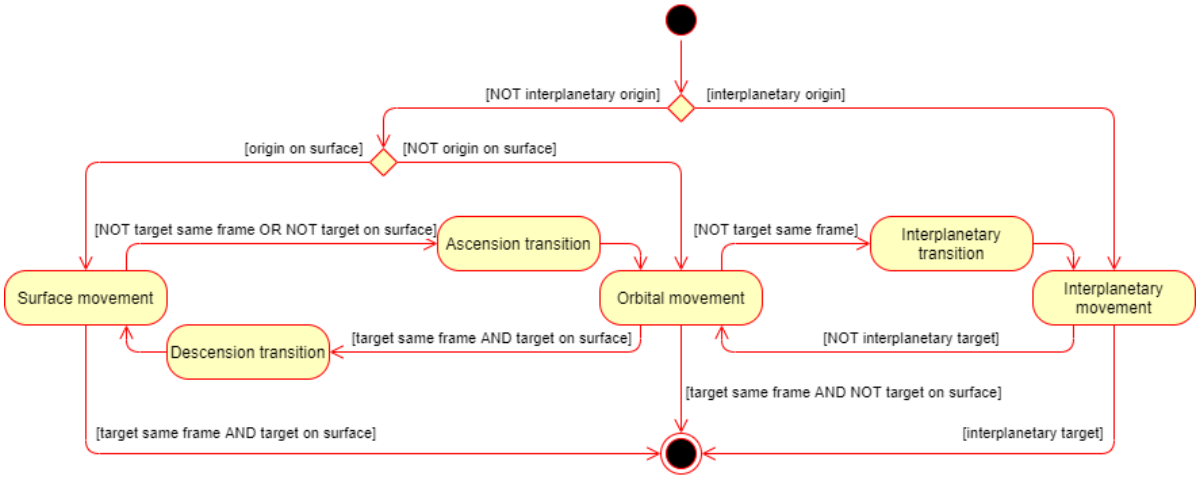



Figure 4.4: State diagram for the different types of movement and the transitions between them.

the different movements are structured in layers, therefore a transition from surface to interplanetary space and vice versa is done via transition from surface to orbit, and from orbit to interplanetary space. Additionally, there is no transition from interplanetary movement to orbital movement since the interplanetary movements end in a position where either the destination is reached, or the observer is in a poition in the target orbit and is able to resume orbital movement immediately.

An example for this layered travel is a movement from a location on the origin body's surface to a location on the target body's surface: the movement starts in the "Surface movement" state, since the origin location is on a body's surface. First, the observer transitions from the surface location to an orbit location via the ascension transition, because the observer is still in the origin's SPICE frame. Then, since the observer is still in the wrong frame, the observer transitions to a position for interplanetary movement, and moves to towards the target via interplanetary movement, ending in an orbital location in the target frame. Then, an orbital movement is performed to move above the final location on the surface. Finally, the observer enters a decension transitions to the final location on the surface.

While a SPICE frame is merely a nested oriented frame of reference, and therefore not limited in distance from its center, in the context of the navigation concept, the frame is used as a general reference to a body's sphere of influence, where the observer is eiter on the surface or in an arbitrarily, but reasonably high orbit around the body.

#### Surface Movement

In this overhaul, we did not primarily focus on a solution for automatic movement on the surface of a body, since those movements are strongly dependent on distance and topology between the target and origin. Additionally, surface navigation is difficult to test under the current circumstances,

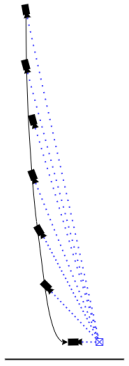


Figure 4.5: Planned

since a connection to the DLR map server is needed to poll street level elevation data and textures, and both were unavailable in the remote work setting due to the COVID-19 pandemic.
In order to find a general solution, independent of distance and topology, we chose to facilitate surface movements via orbital travel, i.e. transitioning from the surface origin location into orbit, using orbital movement to go to an orbital location above the target surface location and transitioning back down to the surface at the target location.
While this solution may be inconvenient for short distances relative to the body's circumference, we assume this form of movement to be a default solution independent of topology or distance. Further solutions for surface movements, especially for short distances, should be easily implementable in later works using the established framework we develop here.

The important changes in the surface movement are the transitions between surface and orbit (the landing and ascending movements).
In order to prevent the change of reference frame, as pointed out in section 3.2.2, we plan to change the linear path and simultaneous rotation shown in figure 3.4, to a parabolic curve, so the observer's movements are always perceived as forward, not downward, and the observer swoops into the target location on the surface, as depicted in figure 4.5.
To ease the rotation and enable the user to predict the movement, the center of the viewport is aimed at a point slightly in front of the final position in direction of the final orientation. This essentially separates the translation from the rotation due to the steepness of the movement. The bulk of the rotation during the movement is at the end of the curve where the curvature is highest.

The ascending transition is basically similar, but in reverse, so the observer zooms out moving backwards and focusing on the point of origin until it reaches the default orbital distance where the point of origin should almost coincide with the body's center.

#### Orbital Movement

The default orbital distance for each body is relative to the body size, so that the orbited body always occupies the same space of the viewport, independent of the body's size. In order to prevent movements through the body as shown in figure 3.5 in section 3.2.2, we plan to use the flexibility of splines to move the observer in circular curves around the body, while focusing the center of the viewport on the center of the body.
While this movement has simultaneous translation and rotation, this movement is not perceived by most users as the observer moving around the body, but a rotation of the whole simulation around the body where the viewport is focused on. In case either the starting point, or the final location are not equidistant to the boday's center, the curve becomes less circular, but the nature of the spline should compensate for different distances to the center automatically, leading to the user perceiving the change in distance as the observer zooming out, away from the body during the rotation.

For the transition from orbital to interplanetary movement, a few steps are added to reduce surprising rotations and make the movement easier to predict by the user.

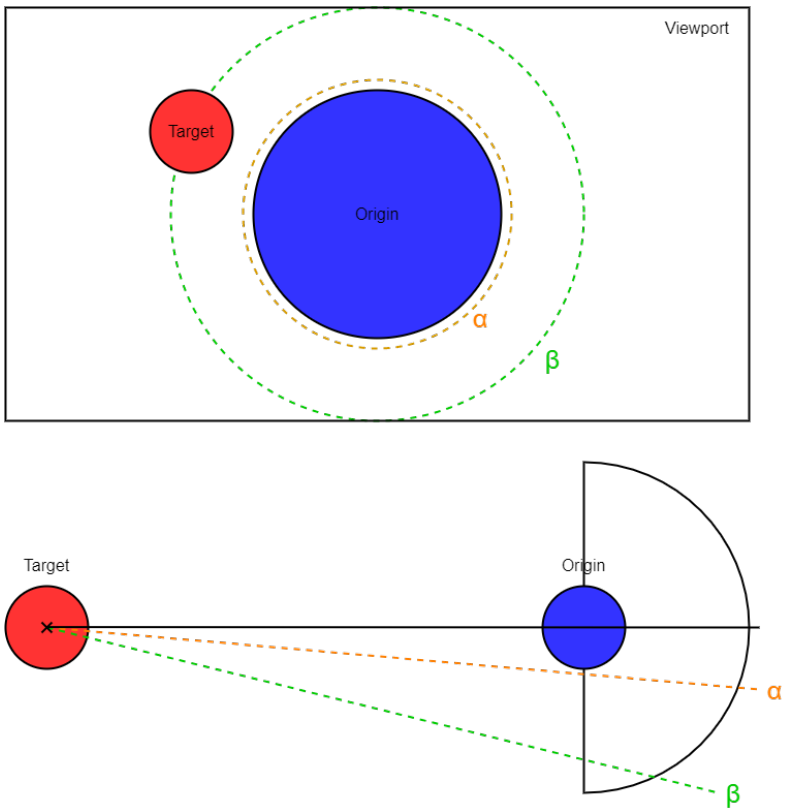



Figure 4.6: Minimum ($\alpha$) and maximum ($\beta$) angle for the transition into interplanetary movement.

First, the observer finds the closest suitable location where the following conditions are met:

- The angle $\theta$ between the vector from the target's center to the observer, and the vector from the target's center to the origin's center is between the minimum allowed angle $\alpha$ and the maximum allowed angle $\beta$.

$$\alpha \geq \theta \geq \beta \tag{4.2}$$

- The vector $\vec{v}$ between the target's center, and the observer must be longer than the vector $\vec{w}$ between the target's center, and the origin's center.

$$\|\vec{v}\| > \|\vec{w}\| \tag{4.3}$$

The requirements are visualised in figure 4.6. The first requirement (equation 4.2), creates two cones from the target's center towards the origin. The $\alpha$-cone is the minimum angle, that prevents the target from being mostly obscured by the origin body, and the $\beta$-cone is the maximum angle, that ensures the Target is still visible on the viewport. Both cones intersect the sphere of the origin's possible orbit locations twice, resulting in two spherical segments.
The second requirement excludes the spherical segment that is in between the origin and target bodies. The spherical zone of the remaining segment are the available locations for the interplanetary transitions, figure 4.7 visualises the possible locations.

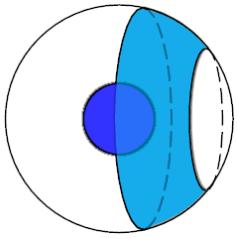


Figure 4.7: Spherical zone of locations available for the interplanetary transition.

The transition for interplanetary movement is to move to the closest location in this spherical zone via orbital movement, which results in a scene similar to the viewport in figure 4.6, where the viewport is focused on the center of the origin's body, and the target body is within the $\alpha$ and $\beta$ margins in the viewport.

#### Interplanetary Movement

The interplanetary movement remains closest to the original movement. However, the interpolation between the origin and target orientation is removed in order to separate rotation from translation and increase predictability. Instead, the rotation is split into an initial rotation to fave the traveling direction, so the user always moves forward, towards the target, and a final rotation at the end of the translation to rotate the observer into the final orientation at the target location.
The translation uses a straight spline. This way the movement is similar to the original interpolation between the origin and target position. However, the spline allows easier modification of the path, leading to easier collision detection and mitigation in the future, since the spline can be easily checked for collisions and, if necessary, additional control points can be inserted into the spline to avoid collisions. The interplanetary movement always ends in either an interplanetary location (bookmark) or a target body's orbit, where orbital movement can be resumed. Therefore, no transition from interplanetary movement to orbital movement is needed.

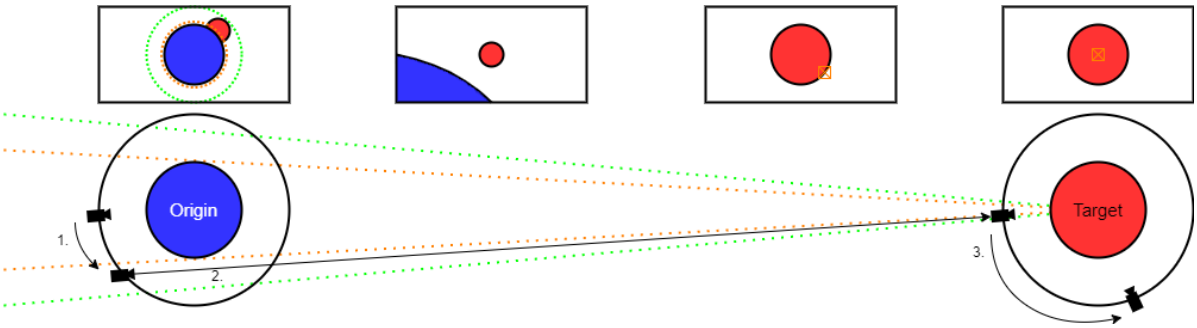



Figure 4.8: Example of automatic movement including the orbital to interplanetary movement transition (1.), the interplanetary movement (2.), and orbital movement the the final position (3.).

An example of movement between two orbital locations, is shown in figure 4.8.
The first step shown in the figure is the transition from an arbitrary orbital location around the origin body to a location that fulfills the requirements (equations 4.2 & 4.3).
The second step is the interplanetary movement between the origin orbit, and the target orbit. First, the observer rotates from focusing on the origin's center to the target's center, and travel direction. During this rotation the requirements for the transition between orbital and interplanetary movement, help reduce the angular distance of the rotation and prevent collision with the origin body during interplanetary travel.
After arriving at the target's orbit, the third step is an orbital movement to the final position.

While this staged approach increases the overall travel time of the automatic navigation, we aim to reduce disorientation and resulting cybersickness symptoms by providing the user with an easily predictable solution that decouples rotation from translation and minimizes complex and surprising rotations.

### 4.3.2 Automatic Movement Implementation

The automatic movement for CosmoScout is organized in two distinct parts. The first part is the *moveTo*-method, where the parameters of the movement are processed, and the second part is the *updateMovementAnimation*-method which is called every frame, polling the state of the movement based on the current time.

The old *moveTo*-method uses the information about the current position and orientation for the origin, as well as the parameter information about the target location, and the duration of the movement to construct two *AnimatedValue* objects.
An *AnimatedValue* object is consists of a start and end value, as well as a start and end time. The object then provides a method to poll the value using time as the parameter, returning either the start or end value, if the time is outside the timespan provided by the start and end time of the *AnimatedValue*, or the method returns an interpolated value between the start and end values relative to the point in time between the start and end time.
One *AnimatedValue* is used for the translation, interpolating between the origin position, and the target position over the duration of the movement. The other *AnimatedValue* is used for the rotation, interpolating between the origin orientation and the target orientation over the duration of the movement. After the *moveTo*-method has processed the available information into the two *AnimatedValue* objects, the *mAnimationInProgress* flag is set. While the flag is true, the *updateMovementAnimation*-method uses the current time at each frame to poll the current position and orientation from the *AnimatedValue* objects, until the time is past the end time of both end times. Afterwards, the *updateMovementAnimation* resets the *mAnimationInProgress* flag to false.

Since the automatic movement can potentially be called from anywhere in CosmoScout, including plugins, we plan to ensure backwards compatibility while changing the way the automatic movement is performed. Therefore, the original *moveTo*-method's signature is kept and additional signatures are added as needed. The original *moveTo*-method was an interpolation between the origin and target location leading to a linear movement path from the origin to the target. Since the interplanetary movement shows similar characteristics, the *moveTo*-method that contains the old method signature is also used for interplanetary travel, and generally linear movements.

The first change to the method is replacing the *AnimatedValue* object for the interpolation of the observers position with a linear spline between the origin and target position, and using the *AnimatedValue* object to interpolate the $t$ parameter over the length of the spline relative to the duration of the movement.
Second, split the rotation into an initial rotation, from the origin orientation to a *mLookAtPoint* in movement direction, and a final rotation, from the *mLookAtPoint* to the final orientation. Additionally, the rotations are separated from the translation by adjusting the start and end times of the *Animated-*

*Value*-objects.
The overall duration ($t_{all}$) is split into three stages, the initial rotation ($t_{start}$), the translative phase ($t_{move}$), and the final rotation ($t_{end}$). For the duration of each rotation a function (equation 4.5) is used that calculates a weight factor ($w$) between 0 and 1 based on the angular difference ($\theta$) of the start and the end direction of view (negative $z$-axis). This way, the durations for each rotation can vary in length based on the angular difference of the rotation, and the translative phase takes up the rest of the duration as described in the following equations:

$$\frac{w_{start}}{3}t_{start} + (1 - \frac{w_{start}}{3} - \frac{w_{end}}{3})t_{move} + \frac{w_{end}}{3}t_{end} = t_{all} \tag{4.4}$$

with:

$$w = f(\theta) = -\frac{\cos\theta - 1}{2} \tag{4.5}$$

This results in the following limitations for the durations:

$$0 \leq t_{start}, t_{end} \leq \tfrac{1}{3}t_{all} \tag{4.6}$$

$$\tfrac{1}{3}t_{all} \leq t_{move} \leq t_{all} \tag{4.7}$$

The modifications to the *moveTo*-method also require changes to the *updateMovementAnimation*-method, as the position and rotation needs to be polled from different sources. Additionally, the simulation is changed to stop the simulation time during movement. The automatic movement is programmed to transform all positions into the target's SPICE frame, and the delay of the initial rotation before translation results in inadvertent movement, as the target frame rotates, moving the observer significantly due to the distance to the target's center.

### Changes to *updateMovementAnimation*

```
1  void CelestialObserver::updateMovementAnimation(double tTime) {
2      if (mAnimationInProgress) {
3          // get position from spline
4          mPosition = mMoveSpline->getPosition(mAnimatedT.get(tTime));
5
6          if (tTime < mAnimatedRotationStart.mEndTime) {
7              // rotation to look-at point not done yet
8              mRotation = mAnimatedRotationStart.get(tTime);
9          } else if (tTime > mAnimatedRotationFinal.mStartTime) {
10             // rotation to final direction started
11             mRotation = mAnimatedRotationFinal.get(tTime);
12         } else {
13             // observer is moving along spline -> adjust rotation towards look-at point
14             auto direction = glm::normalize(mLookAtPoint - mPosition);
15             mRotation      = glm::quatLookAt(direction, mUpDirection);
16         }
17
18         if (mAnimatedRotationFinal.mEndTime < tTime) {
19             mAnimationInProgress = false;
20             if (!mMovementQueue.empty()) {
21                 moveTo(mMovementQueue.front());
```

```
22                      mMovementQueue.pop();
23                  }
24              }
25          }
26      }
```

The polling of the current position only has minimal changes (l.4), as the *AnimatedValue* is polled to receive the *t* parameter for the position on the spline based on the current time, instead of receiving the interpolated position directly from the *AnimatedValue*.
Polling the orientation is dependent on the current time in the movement process (ll.6–16). If the current time is before the end time of the initial rotation, the current orientation is provided by the *AnimatedValue* of the initial rotation, interpolating between the original orientation, and the orientation towards the *mLookAtPoint*. If the current time is after the start time of the final rotation, the orientation is provided by the *AnimatedValue* of the final rotation, interpolating between the orientation aimed at the *mLookAtPoint*, and the final orientation. Otherwise, the movement animation is in the translative phase, and the current orientation is calculated using the *mLookAtPoint*, and the *mUpDirection* to generate a new orientation focused on the *mLookAtPoint*. The *mUpDirection* is saved from the orientation of the observer at the beginning of the movement animation, to prevent the camera from any rolling rotations during the movement animation, since rolling during movements appears to increase disorientation and therefore discomfort in users. Lines 18–24 of the code block contain another important change that is explained in the next section.

#### Movement description packages and Movement Queue

Based on the new changes to the *updateMovementAnimation*-method, there are the following requirements any movement must fill with data in order for the movement animation to display correctly:

- The *mMoveSpline*, providing the path for the movement, and the *AnimatedValue* for the interpolation of the *t* parameter for the spline.
- The *AnimatedValue* for the initial rotation from the origin direction towards the *mLookAtPoint*.
- The *mLookAtPoint* to focus on during the translation, and the *mUpDirection* to calculate the orientation and prevent rolling.
- The *AnimatedValue* for the final rotation from the direction towards the *mLookAtPoint* to the final direction.

Any new movement must provide these information and therefore, a movement always consist of the initial rotation, then the translation with a given point focused in the center of the viewport during the movement, and a final rotation into the final orientation.
To provide functionality for the transitions, as well as future, potentially complex movements and routes, a queue is added to the automatic movement system. The parameters needed by the *moveTo*-methods to process into the above mentioned requirements are packaged into structs, and the *mMovementQueue* provides a FIFO queue to hold an ordered list of these movement instructions. At the end of the *updateMovementAnimation*-method (ll.18–24), the *mMovementQueue* is checked whether it contains any items, and if so, the topmost item is removed, and the *moveTo*-method is called with the struct as parameter. For this additional *moveTo*-methods can be added to provide different types of movement.

Additionally, a *moveTo*-method is added that accepts a queue of movement instructions as a parameter, as shown below:

```
1  void CelestialObserver::moveTo(std::queue<MovementDescription> const& moveDescriptionsQueue) {
2      // make sure queue contains at least one element
3      if (!moveDescriptionsQueue.empty()) {
4          // swap new movements into queue
5          mMovementQueue = moveDescriptionsQueue;
6          // execute first movement instruction
7          moveTo(mMovementQueue.front());
8          mMovementQueue.pop();
9      }
10 }
```

Providing a new, non-empty queue to the method during an ongoing movement discards remaining items, emplaces the new queue into the *mMovementQueue*, and calls the *moveTo*-method with the topmost item of the queue. Discarding the old queue when receiving a new queue is done, to prevent buffering of multiple, potentially unwanted movements. Currently, the *mMovementQueue* can be filled with a variant of one of the two basic movement description packages. Either the linear movement that resulted from the original *moveTo*-method, and is used in the interplanetary movement, or the circular movement that is used in the orbital movement.

#### Different *moveTo*-methods and control point computation

To provide different types of movement, the *moveTo*-method has multiple implementation with different signatures:

```
1  void moveTo(std::string const& sCenterName, std::string const& sFrameName,
2      glm::dvec3 const& position, glm::dquat const& rotation, double dSimulationTime,
3      double dRealStartTime, double dRealEndTime);
4
5  [...]
6
7  void moveTo(defaultPoint2Point const& moveDescriptionP2P);
8
9  [...]
10
11 void moveTo(defaultOrbit const& moveDescriptionOrbit);
12
13 [...]
14
15 void moveTo(MovementDescription const& moveDescription);
16
17 [...]
18
19 void moveTo(std::queue<MovementDescription>const & moveDescriptionsQueue);
```

The original signature of the method (l.1) is kept to allow backwards compatibility. The second signature (l.7) accepts movement description packages for linear, point-to-point (interplanetary) movements. The function simply unpacks and relays the parameters for the linear movement to the first *moveTo*-method. The third signature (l.11) accepts movement description packages for circular (orbital) movements. The fourth signature (l.15) is added because the movement queue uses a variant type for its elements.

Therefore, when a new element for the queue is used, this method accepts it. The method checks the real movement description type using a switch-case-statement, and typecasts the variant-type elements, and relays them to their respective methods. The last signature (l.19) accepts whole movement queues to be passed and is described above. Currently, only linear and circular movement packages are implemented, however, the addition of further movement description packages is straightforward in this system.

The movement path is constructed using a uniform cubic basis splines. Uniform cubic b-splines offer some advantages over other curves, as they are similar to bezier curves, which means they are relatively easy to construct, and have a continuous velocity over the curve. Additionally, uniform cubic b-splines offer a continuous curvature, easing the movement of the observer along the spline. Lastly, their control points only have local influence on the curve, which means that additional control points can be inserted to handle collisions, without major changes to the overall curve. Their only disadvantage, the interpolated curve not necessarily passing through the specified control points, can be mitigated by inserting triplets of control points with the middle point being the control point teh curve should pass close to, and the other two points representing a tangent to the curve that is passing through the three control points.
Uniform cubic b-splines that are non-looping also require an additional point on either end of the curve where the curve is not interpolated. Therefore, the curve for the movement always starts in the second and ends in the penultimate control point.

The spline for the interplanetary movement is constructed between the origin and target using two triplets of control points.


Figure 4.9: Control points of a linear spline with the start and end point ($P_{start}$, $P_{end}$), and the control points ($P_{S-OT}$, $P_{S+OT}$, $P_{E-OT}$, $P_{E+OT}$) making up each triplet.

First, the vector between the origin and target ($OT$-vector) is constructed and normalized. Then, each triplet forms the tangent through the start and end point, where the start or end point is the middle control point, with one control point in front, and one control point behind the middle point in direction of the $OT$-vector, as described in figure 4.9. This way, bot the tangent in the start and end point are along the $OT$-vector, leading to a straight spline between the start and end point. The resulting spline is saved in the *mMoveSpline*, and the *AnimatedValue*-object for the interpolation of the $t$ parameter is set to interpolate between 0 and $t_{max}$ (1 in most cases) over the duration of the translation phase.
The *mLookAtPoint* is set to the last control point, as it is slightly in front of the end point of the interpolated spline. This way the observer is always oriented towards the end point of the movement path, without the movement ending in a location where the position and *mLookAtPoint* coincide, which could lead to an undefined orientation for the observer. Additionally, the *mUpDirection* is saved from the up-direction of the observer in the origin location to prevent the observer from rolling during the movement phase.
Finally, the *AnimatedValue*-objects for the initial, and the final rotation are set. The initial rotation interpolates between the origin orientation, and the orientation towards the *mLookAtPoint* over the

duration for the initial rotation. The final rotation interpolates between the orientation towards the *mLookAtPoint* and the final orientation over the duration for the final rotation.
The durations and points in time for the rotations and the translation are calculated as mentioned above (equations 4.5 and 4.4).

The orbital movement uses a different *moveTo*-method to realize the circular movement path. To approximate the circular arc between the start and end point, both points are projected into 2D space by finding the normal to the plane defined by the center-start, and center-end vectors ($\overrightarrow{cs}$ and $\overrightarrow{ce}$). This also results in the shortest possible path in orbit from the start to the end point. Next, the angle ($\theta$) between the two vectors is calculated, and both vectors are normalized to compute the additional control points on the unit circle.

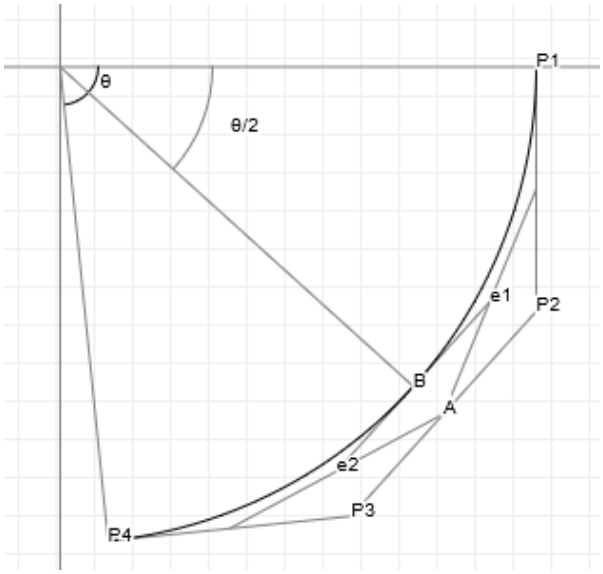



Figure 4.10: Control points ($P2$, $P3$), the start point ($P1$), end point ($P4$) and angle ($\theta$) in 2D space [57].

Figure 4.10 shows an example of the transformation into 2D space, and the required control points ($P1$, $P2$, $P3$, and $P4$). Poxmax [57] derived a formula to calculate the missing control points $P2$ and $P3$ on the unit circle:

$$\begin{aligned} P_{start} &= P1 = (1, 0) \\ P_{c1} &= P2 = (1, k) \\ P_{c2} &= P3 = (\cos(\theta) + k \cdot \sin(\theta), \sin(\theta) - k \cdot \cos(\theta)) \\ P_{end} &= P4 = (\cos(\theta), \sin(\theta)) \end{aligned} \tag{4.8}$$

With:

$$k = f(\theta) = \frac{4}{3} \cdot \tan\left(\frac{\theta}{4}\right) \tag{4.9}$$

To transform the control points back into 3D space, the oriented angle around the center from $P1$ to $P2$ ($\phi_{start}$), and from $P4$ to $P3$ ($\phi_{end}$) is calculated. To get the control points, the start position and end position are rotated around the normal of the 2D plane, and scaled by the magnitude of their 2D counterpart.

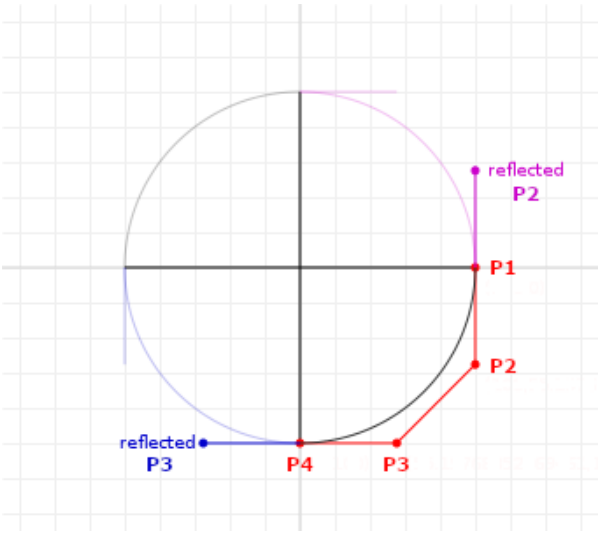



Figure 4.11: Example control points for circular spline where $\theta = 90$ deg [57].

Since, the spline needs two additional control points where the spline is not interpolated, the calculated control points are also reflected to the other side of the start and end position as shown in figure 4.11. The control points in 3D space ($C_{P2}$, $C_{P3}$), and their reflections ($C_{rP2}$, $C_{rP3}$) can be calculated using Rodrigues' rotation formula with the start and end points ($C_{start}$, $C_{end}$), the normal for the 2D plane ($\vec{N}$), the angles between the control point, and the start, or end point ($\phi_{start}$, $\phi_{end}$), and the control points in 2D space ($\overrightarrow{P2}$, $\overrightarrow{P3}$):

$$
\begin{aligned}
C_{P2} &= \left(\vec{C}_{start}\cos(\phi_{start}) + (\vec{N}\times\vec{C}_{start})\sin(\phi_{start}) + \vec{N}(\vec{N}\cdot\vec{C}_{start})(1-\cos(\phi_{start}))\right)\cdot\left\|\overrightarrow{P2}\right\| \\
C_{rP2} &= \left(\vec{C}_{start}\cos(-\phi_{start}) + (\vec{N}\times\vec{C}_{start})\sin(-\phi_{start}) + \vec{N}(\vec{N}\cdot\vec{C}_{start})(1-\cos(-\phi_{start}))\right)\cdot\left\|\overrightarrow{P2}\right\| \\
C_{P3} &= \left(\vec{C}_{end}\cos(\phi_{end}) + (\vec{N}\times\vec{C}_{end})\sin(\phi_{end}) + \vec{N}(\vec{N}\cdot\vec{C}_{end})(1-\cos(\phi_{end}))\right)\cdot\left\|\overrightarrow{P3}\right\| \\
C_{rP3} &= \left(\vec{C}_{end}\cos(-\phi_{end}) + (\vec{N}\times\vec{C}_{end})\sin(-\phi_{end}) + \vec{N}(\vec{N}\cdot\vec{C}_{end})(1-\cos(-\phi_{end}))\right)\cdot\left\|\overrightarrow{P3}\right\|
\end{aligned}
\tag{4.10}
$$

Since the start and end position are used to derive the adjacent control points, different orbit heights in start and end position are carried over to their adjacent control points, keeping the triplet of control points on the tangent to the movement curve in the start or end position. Differences in orbit height between the start triplet and end triplet lead to the movement spline becoming more elliptical, resulting in a zoom effect from the start to the end orbit height during the rotation around the center.
With the control points calculated, the *mMoveSpline* can be constructed, and the *AnimatedValue*-object for the $t$ parameter can be set to interpolate the position along the spline over the duration of the movement phase. The *mLookAtPoint* for the orientation during the translation phase is set to the center of the body, and the *mUpDirection* is saved from the origin orientation to prevent rolling during the movement. The *AnimatedValue*-object for the initial rotation is set to interpolate from the origin orientation to the orientation facing the center of the body over the duration of the initial rotation. The *AnimatedValue*-object for the final rotation is set to interpolate from the orientation facing the center of the body to the final orientation over the duration of the final rotation. The durations for the two rotation

are calculated in the same way as the durations for the linear movement, based on the angular difference between the start and end direction. However, to properly calculate the angular difference, roll rotation has to be taken into account as well, therefore the difference between the start and end quaternion is used to calculate the weight instead of the equation 4.5.

# 5 User Study

# 6 Further Work

# A Appendix

This is the appendix. You can put all the stuff you like here.

## A.1 Appendix Sections

The enumeration for the appendix is different.

# Bibliography


[1] Rebenitsch, L. and C. Owen: *Review on cybersickness in applications and visual displays*. In *Virtual Reality*, vol. 20, pp. 101–125, 2016. https://link.springer.com/article/10.1007%2Fs10055-016-0285-9.

[2] LaViola, J. J.: *A discussion of cybersickness in virtual environments*. SIGCHI Bull., 32(1):47–56, 1 2000. https://doi.org/10.1145/333329.333344.

[3] Post, R. E. and L. M. Dickerson: *Dizziness: a diagnostic approach*. American family physician, 82(4):361–368, 8 2010. https://www.aafp.org/afp/2010/0815/p361.html, PMID: 20704166.

[4] Ruddle, R. A.: *Colorplate: The effect of environment characteristics and user interaction on levels of virtual environment sickness*. In *Virtual Reality Conference, IEEE*, vol. 1, p. 285, Los Alamitos, CA, USA, mar 2004. IEEE Computer Society. https://doi.ieeecomputersociety.org/10.1109/VR.2004.10029.

[5] Min, B. C., S. C. Chung, Y. K. Min, and K. Sakamoto: *Psychophysiological evaluation of simulator sickness evoked by a graphic simulator*. Applied Ergonomics, 35(6):549–556, 2004. https://www.sciencedirect.com/science/article/pii/S0003687004000985.

[6] Dużmańska, N., P. Strojny, and A. Strojny: *Can simulator sickness be avoided? a review on temporal aspects of simulator sickness*. Frontiers in Psychology, 9:2132, 2018. https://www.frontiersin.org/article/10.3389/fpsyg.2018.02132.

[7] Hill, K. J. and P. Howarth: *Habituation to the side effects of immersion in a virtual environment*. Displays, 21(1):25–30, 2000. https://www.sciencedirect.com/science/article/pii/S0141938200000299.

[8] Saredakis, D., A. Szpak, B. Birckhead, H. A. D. Keage, A. Rizzo, and T. Loetscher: *Factors associated with virtual reality sickness in head-mounted displays: A systematic review and meta-analysis*. Frontiers in Human Neuroscience, 14:96, 2020. https://www.frontiersin.org/articles/10.3389/fnhum.2020.00096/full.

[9] Kennedy, R. S., N. E. Lane, K. S. Berbaum, and M. G. Lilienthal: *Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness*. The International Journal of Aviation Psychology, pp. 203–220, 1993. https://doi.org/10.1207/s15327108ijap0303_3.

[10] McCauley, M. E. and T. J. Sharkey: *Cybersickness: Perception of self-motion in virtual environments*. In *Presence: Virtual and Augmented Reality*, vol. 1, pp. 311–318, 1992. https://doi.org/10.1162/pres.1992.1.3.311.

[11] Kim, H. K., J. Park, Y. Choi, and M. Choe: *Virtual reality sickness questionnaire (vrsq): Motion sickness measurement index in a virtual reality environment*. Applied Ergonomics, 69:66–73, 2018. https://www.sciencedirect.com/science/article/pii/S000368701730282X.

[12] Cobb, S. V. G., S. Nichols, A. Ramsey, and J. R. Wilson: *Virtual reality-induced symptoms and effects (vrise)*. In *Presence: Virtual and Augmented Reality*, vol. 8, pp. 169–186, 1999. https://doi.org/10.1162/105474699566152.

[13] Stanney, K. M., R. S. Kennedy, and J. M. Drexler: *Cybersickness is not simulator sickness*. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 41, pp. 1138–1142, 10 1997. https://doi.org/10.1177/107118139704100292.

[14] Weech, S., J. Moon, and N. F. Troje: *Influence of bone-conducted vibration on simulator sickness in virtual reality*. PLOS ONE, 13(3):1–21, 03 2018. https://doi.org/10.1371/journal.pone.0194137.

[15] Keshavarz, B., A. E. Philipp-Muller, W. Hemmerich, B. E. Riecke, and J. L. Campos: *The effect of visual motion stimulus characteristics on vection and visually induced motion sickness*. Displays, 58:71–81, 2019. https://www.sciencedirect.com/science/article/pii/S0141938218301112.

[16] Palmisano, S., R. Mursic, and J. Kim: *Vection and cybersickness generated by head-and-display motion in the oculus rift*. Displays, 46:1–8, 2017. https://www.sciencedirect.com/science/article/pii/S0141938216300713.

[17] Walker, M.: *Vestibular system*. In Aminoff, M. J. and R. B. Daroff (eds.): *Encyclopedia of the Neurological Sciences (Second Edition)*, pp. 647–656. Academic Press, Oxford, second edition ed., 2014. https://www.sciencedirect.com/science/article/pii/B9780123851574011854.

[18] Barrett, G. V. and C. L. Thornton: *Relationship between perceptual style and simulator sickness.* Journal of Applied Psychology, 52(4):304–308, 1968. https://doi.org/10.1037/h0026013.

[19] Tiiro, A.: *Effect of visual realism on cybersickness in virtual reality*, 2018. http://jultika.oulu.fi/files/nbnfioulu-201802091218.pdf.

[20] Kolasinski, E. M.: *Simulator Sickness in Virtual Environments*. Technical report (U.S. Army Research Institute for the Behavioral and Social Sciences). U.S. Army Research Institute for the Behavioral and Social Sciences, 1995. https://books.google.de/books?id=7qwrAAAAYAAJ.

[21] Riccio, G. E. and T. A. Stoffregen: *An ecological theory of motion sickness and postural instability*. Ecological Psychology, 3(3):195–240, 1991. https://doi.org/10.1207/s15326969eco0303_2.

[22] Clifton, J. and S. Palmisano: *Effects of steering locomotion and teleporting on cybersickness and presence in hmd-based virtual reality*. Virtual Reality, 24(3):453–468, Sep 2020. https://doi.org/10.1007/s10055-019-00407-8.

[23] Stoffregen, T. A. and L. J. Smart: *Postural instability precedes motion sickness*. Brain Research Bulletin, 47(5):437–448, 1998. https://www.sciencedirect.com/science/article/pii/S0361923098001026.

[24] Smart, L. J., E. W. Otten, H. E. Cook IV, A. J. Kinesella, L. E. Sullivan, L. R. Amin, and J. L. Braun: *The sickness profile: Characterizing postural instability*, 11 2013. https://www.researchgate.net/publication/258258363_The_Sickness_Profile_Characterizing_Postural_Instability.

[25] Chardonnet, J. R., M. A. Mirzaei, and F. Merienne: *Visually induced motion sickness estimation and prediction in virtual reality using frequency components analysis of postural sway signal*. In *International Conference on Artificial Reality and Telexistence Eurographics Symposium on Virtual Environments*, pp. 9–16, Kyoto, Japan, 10 2015. https://hal.archives-ouvertes.fr/hal-01229880.

[26] Lim, K., J. Lee, K. Won, N. Kala, and T. Lee: *A novel method for vr sickness reduction based on dynamic field of view processing*. Virtual Reality, 7 2020. https://doi.org/10.1007/s10055-020-00457-3.

[27] Chang, E., I. Hwang, H. Jeon, Y. Chun, H. T. Kim, and C. Park: *Effects of rest frames on cybersickness and oscillatory brain activity*. pp. 62–64, 2 2013. https://doi.org/10.1109/IWW-BCI.2013.6506631.

[28] Duh, H. B. L., D. E. Parker, and T. A. Furness: *An "independent visual background" reduced balance disturbance evoked by visual scene motion: Implication for alleviating simulator sickness*. pp. 85–89, 01 2001. https://doi.org/10.1145/365024.365051.

[29] Kroeker, K. L.: *Looking beyond stereoscopic 3d's revival*. Commun. ACM, 53(8):14–16, 8 2010. https://doi.org/10.1145/1787234.1787241.

[30] Kim, J., D. Kane, and M. S. Banks: *The rate of change of vergence–accommodation conflict affects visual discomfort*. Vision Research, 105:159–165, 2014. https://www.sciencedirect.com/science/article/pii/S0042698914002545.

[31] Ames, S. L., J. S. Wolffsohn, and N. A. McBrien: *The development of a symptom questionnaire for assessing virtual reality viewing using a head-mounted display*. Optometry and Vision Science, 82(3):168–176, 2005. https://doi.org/10.1097/01.OPX.0000156307.95086.6.

[32] Stone III, W. B.: *Psychometric evaluation of the Simulator Sickness Questionnaire as a measure of cybersickness*. PhD thesis, Iowa State University, 2017. https://lib.dr.iastate.edu/etd/15429.

[33] Gianaros, P. J., E. R. Muth, J. T. Mordkoff, M. E. Levine, and R. M. Stern: *A questionnaire for the assessment of the multiple dimensions of motion sickness*. Aviat Space Environ Med, 72(2):115–119, 2 2001. https://www.ncbi.nlm.nih.gov/pmc/articles/pmid/11211039/, PMID: 11211039.

[34] Keshavarz, B. and H. Hecht: *Validating an efficient method to quantify motion sickness*. Human Factors, 53(4):415–426, 2011. https://doi.org/10.1177/0018720811403736, PMID: 21901938.

[35] Kim, Y. Y., H. J. Kim, E. N. Kim, H. D. Ko, and H. T. Kim: *Characteristic changes in the physiological components of cybersickness*. Psychophysiology, 42(5):616–625, 2005. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8986.2005.00349.x.

[36] Sevinc, V. and M. I. Berkman: *Psychometric evaluation of simulator sickness questionnaire and its variants as a measure of cybersickness in consumer virtual environments*. Applied Ergonomics, 82, 2020. https://www.sciencedirect.com/science/article/pii/S0003687019301759.

[37] Davis, S., K. Nesbitt, and E. Nalivaiko: *A systematic review of cybersickness*. In *Proceedings of the 2014 Conference on Interactive Entertainment*, p. 1–9, New York, NY, USA, 2014. Association for Computing Machinery. https://doi.org/10.1145/2677758.2677780.

[38] Graybiel, A. and J. Knepton: *Sopite syndrome: a sometimes sole manifestation of motion sickness*. Aviat Space Environ Med, 47(8):873–882, 8 1976. https://pubmed.ncbi.nlm.nih.gov/949309/, PMID: 949309.

[39] Roberts, W. K. and J. J. Gallimore: *A physiological model of cybersickness during virtual environment interaction*. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 49(26):2230–2234, 2005. https://doi.org/10.1177/154193120504902603.

[40] Kiryu, T., E. Uchiyama, M. Jimbo, and A. Iijima: *Time-varying factors model with different timescales for studying cybersickness*. ICVR 2007: Virtual Reality, 4563:262–269, 2007. https://doi.org/10.1007/978-3-540-73335-5_29.

[41] Watanabe, H. and H. Ujike: *The activity of iso/study group on image safety and three biological effect*. pp. 210 – 214, 2008. https://doi.org/10.1109/ISUC.2008.11.

[42] Villard, S. J., M. B. Flanagan, G. M. Albanese, and T. A. Stoffregen: *Postural instability and motion sickness in a virtual moving room*. Human Factors, 50(2):332–345, 2008. https://doi.org/10.1518/001872008X250728, PMID: 18516843.

[43] Dong, X., K. Yoshida, and T. A. Stoffregen: *Control of a virtual vehicle influences postural activity and motion sickness*. Journal of Experimental Psychology: Applied, 17(2):128–138, 6 2011. https://doi.apa.org/doi/10.1037/a0024097.

[44] Dong, X. and T. A. Stoffregen: *Postural activity and motion sickness among drivers and passengers in a console video game*. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 54(18):1340–1344, 2010. https://doi.org/10.1177/154193121005401808.

[45] Buhler, H., S. Misztal, and J. Schild: *Reducing vr sickness through peripheral visual effects*. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 517–9, 2018. https://doi.org/10.1109/VR.2018.8446346.

[46] Weech, S., S. Kenny, and M. Barnett-Cowan: *Presence and cybersickness in virtual reality are negatively related: A review*. Front Psychol, 10, 2019. https://doi.org/10.3389/fpsyg.2019.00158, PMID: 30778320.

[47] Duh, H. B. L., J. J. W. Lin, R. V. Kenyon, D. E. Parker, and T. A. Furness: *Effects of field of view on balance in an immersive environment*. In *Proceedings IEEE Virtual Reality 2001*, pp. 235–240, 2001. https://ieeexplore.ieee.org/document/913791.

[48] Lin, J. J. W., H. B. L. Duh, D. E. Parker, H. Abi-Rached, and T. A. Furness: *Effects of field of view on presence, enjoyment, memory, and simulator sickness in a virtual environment*. In *Proceedings IEEE Virtual Reality 2002*, pp. 164–171, 2002. https://ieeexplore.ieee.org/document/996519.

[49] Fernandes, A. S. and S. K. Feiner: *Combating vr sickness through subtle dynamic field-of-view modification*. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 201–210, 2016. https://doi.org/10.1109/3DUI.2016.7460053.

[50] Kemeny, A., P. George, F. Merienne, and F. Colombet: *New vr navigation techniques to reduce cybersickness*. Electronic Imaging, (3):48–53, 2017. https://doi.org/10.2352/ISSN.2470-1173.2017.3.ERVR-097.

[51] Kato, K. and S. Kitazaki: *Improvement of ease of viewing images on an in-vehicle display and reduction of carsickness*. In *SAE Technical Paper*, 4 2008. https://doi.org/10.4271/2008-01-0565.

[52] B A A: *htc vive 3 touchpad*. https://bahfvr.blogspot.com/2017/09/htc-vive-3-touchpad.html.

[53] Drogemuller, A., A. Cunningham, J. Walsh, B. H. Thomas, M. Cordeil, and W. Ross: *Examining virtual reality navigation techniques for 3d network visualisations*. Journal of Computer Languages, 56, 2020. https://doi.org/10.1016/j.cola.2019.100937.

[54] Fragaszy, D. M. and M. Mangalam: *Chapter five - tooling*. Vol. 50 of *Advances in the Study of Behavior*, pp. 177–241. Academic Press, 2018. https://www.sciencedirect.com/science/article/pii/S0065345418300019.

[55] Keshavarz, B. and H. Hecht: *Axis rotation and visually induced motion sickness: The role of combined roll, pitch, and yaw motion*. Aviation, Space, and Environmental Medicine, 82(11):1023–1029, 2011. https://doi.org/10.3357/ASEM.3078.2011.

[56] Hatada, T., H. Sakata, and H. Kusaka: *Psychophysical analysis of the "sensation of reality" induced by a visual wide-field display*. SMPTE Journal, 89(8):560–569, 1980. https://doi.org/10.5594/J01582.

[57] Poxmax: *A primer on bézier curves*. https://pomax.github.io/bezierinfo/#circles_cubic.