Human-Computer Interaction

Department of Computer Science
University of Hamburg

Face Detection and Head Tracking for perspective rendering

Bachelor Thesis in Computer Science

Author:
Jim Martens

Matriculation Number: 6420323

Supervisor: Prof. Dr. Frank Steinicke
Co-Supervisor: Dr. Gerd Bruder

Hamburg, July 19, 2016

License

Acknowledgements

I would like to thank for the continued support, suggestions and advise from my supervisor Prof. Dr. Frank Steinicke, co-supervisor Dr. Gerd Bruder and Susanne Schmidt.

In addition this thesis would not have been possible without the various people I met over the course of my study who provided a friendly environment and the political and societal events that offered distraction and kept me going forward. Especially the engagement in the student council and student parliament allowed me to put my thesis in the greater picture and therefore develop the motivation necessary to finish it.

Last but not least I want to thank the orientation unit for providing me with a good start to my studies. Without it I'd be a different person and I would not have finished my study.

Abstract

Windows are an integral part of our life. They help to reduce stress if they show nature scenes. Yet there are many places without windows or only urban structures. Virtual windows can be used to offer a window-like experience where there is none. This thesis explores the difference between full (real window behaviour) and simple (visible part of world moves but objects don't move relative to each other) motion parallax as well as the impact of steroscopic 3D on the window experience.

The results indicate that for the most part stereoscopic 3D doesn't offer a significant improvement and therefore shouldn't be used as it is more difficult to set up. Full motion parallax delivers the better results in all tested situations but the difference is not significant enough to make it worthwile to switch to complete full motion parallax immediately as that is still very expensive.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Windows are an ancient thing. They already existed – in different facets – over 2,000 years ago. In this timeframe many things have changed but the functionality of a window remained the same. It is generally used to look at things that would otherwise be occluded by a wall. Most windows can be opened and therefore fulfill other functionalities as well (e.g. emergency exits, fresh air supply). They are also a structural weakness which led to the invention of more and more sophisticated windows to keep the cold out.

If robots ruled the world windows would be one of the things to go first. For a machine there are no advantages to a window (list not exhaustive):

- structural weakness
- reduces storage space
- nothing can be mounted to a window

Humans however require things like daylight and can suffer from claustrophobia. Being in a room without windows certainly doesn't help there. In contrast a window allows the eye to escape the physical confines of the room.

Most of the day you use windows in their role as windows to the world. This is also the biggest difference to any image – be it painted or photographed – of the world. By changing your perspective you see different things through the window. You neither need heavy equipment nor an intensive learning process to achieve this. It is quite natural.

That said it seems strange that most visual interaction in our digital age behaves like an image: your perspective doesn't matter. Even 3D movies don't change this. They give depth perception but the view of the world is recorded and equal for every viewer.

This leads to the question if the natural interaction with windows could be replicated with nowadays hard- and software. Short answer: Yes, but with limitations. The concept of having a display working like a window is called "Virtual Window". Essentially it works like this:

A virtual world is shown on the display. This can be a high-resolution photograph, a high-resolution video from one point of view (POV) or a world rendered in real time by a game engine. The shown part of the world changes relative to the perspective of the viewer, giving the feeling of actually looking through a window (gross simplification). The actual feeling depends on many factors.

On the technical side head tracking is required to correctly update the visible frame of the world. Face detection may be used to further enhance this.

Obviously hardware is required to gather the data then used for these techniques. Such hardware exists for some time now but most systems (e.g. motion capture) require a massive set up and are therefore not eligible for the virtual window usage. Even though the effect might be achieved, it won't feel natural if you need to wear for example a mocap suit.

The release of the Microsoft Kinect in 2012[1] changed the landscape in that regard. As it is easy to set up and relatively cheap while still providing good performance and quality, virtual windows became much more feasible.

In the next chapter the previous work will be showcased, followed by a chapter focusing on the Kinect sensor. A chapter will cover the contribution of this thesis including the technical implementation and study related sections. Finally an outlook chapter will look into the future.

Chapter 2

Background

Like every other scientific work nowadays this thesis depends on previous research results. It is part of good behaviour and helps the understanding of the reader to reference the work you have benefitted from (just like in software development). However it doesn't make sense to go back to the invention of the computer. Therefore the focus of this chapter will be on the previous work most relevant to the work of this thesis.

2.1 Usefulness of Virtual Windows

The importance of windows has been shown in multiple studies in the past. Finnegan and Solomon [2] showed that windows are important in offices. Kaplan [3] did so for the home and Ulrich [4] for hospital rooms. These studies have also shown that nature scenes have a positive effect on the wellbeing of humans.

More specifically Finnegan and Solomon [2] used a 19-item questionnaire to gather the data in their study. Each item consisted of a posive or negative statement regarding job satisfaction and a five point agree/disagree scale. The items were grouped in "logical" factors: job satisfaction, interest value, time sense, space sense, anxiety and physical working conditions. 110 female and 13 male subjects completed the questionnaire. 81 worked in windowed settings and 32 in windowless settings. The results for time sense, space sense and anxiety didn't differ significantly between windowless and windowed workers. However the windowless employees were significantly less positive on job satisfaction, interest value of the job, physical working conditions and total results than windowed employees (t values = 2.09, 2.21, 4.10, 2.98 respectively, df = 121).

Kaplan [3] used a questionnaire with mostly five point scales where 1 was the worst and 5 the best value (e.g. very good or very often). It was sent by mail to all residents of some selected residences. In the verbal description part they had to consider how dominant each of 17 characteristics were on the aforementioned five point scales. These characteristics consisted of nature content and built elements that could be visible from the window. The instructions indicated that a parking lot which is mostly covered by a closed curtain would be 2, which means it is seen only occasionally. Another part of the questionnaire were photographs. These were taken from the six buildings and included

all possible window views. The residents were asked to specify how close an image came to their own view using a 5-point scale. In addition they had to specify how much they would like each view to be their own (again using a 5-point scale).

With increasing urbanization it becomes increasingly difficult to find workplaces or homes with windows showing nature scenes. This trend is combined with an ever increasing amount of work and therefore higher stress levels. Windows with nature scenes serve as a countering effect by reducing this stress. The increasing lack of workplaces and homes with windows showing nature scenes therefore presents a major problem to our society.

Virtual windows could come to the rescue. They offer the opportunity to have windows in places without them (e.g. intensive care, some bunker-like university buildings, underground rooms) and they can show nature scenes where only urban structures are in sight. This describes their potential. The actual success heavily depends on the implementation and how well that implementation is received. In short: Solutions must provide high levels of immersion.

Lassonde, Gloth, and Borchert [5] showed that virtual windows help the attention in an otherwise windowless classroom.

2.2 Historic development of Virtual Windows

The idea of virtual windows – or generally illusion – is not new. It was already used 2,000 years ago. Of course not with head tracking and displays but with painting. So called trompe l'oeil paintings make use of a fixed perspective in which the painting looks like a 3D environment. They were used for fake windows and various other things. This painting technique is still used today. Nowadays it is used among other things for street art: An artist paints an image onto the street (e.g. a lake) and from a specific point of view it gives the illusion of being real.

On the technical side commercial applications for virtual windows exist for some time now. Therapeutic Environmental Solutions [6] for example offer virtual window solutions for hospitals to make use of the previously described positive effect of nature scenes on the wellbeing of patients and humans in general. Perrin Photographic Art [7] build virtual windows with a static image, LEDs lighting the image and an image frame providing depth. The focus is here on homes instead of hospitals. Sky Factory [8] develops nature inspired virtual windows and skylights for walls and ceilings. These windows and skylights are used for example in hospital rooms, hotels and workplaces. Häkkilä, Koskenranta, Posti, *et al.* [9] describe a solution where the user can interact with the virtual window by melting the displayed snow with a hand gesture.

All of these solutions have in common that they don't respond to the viewer's position.

It simply doesn't matter how close or far away the viewer is, the presented image/video is always the same for everyone.

Is that a problem though? IJsselsteijn, Oosting, Vogels, *et al.* [10] researched the impact of the cues motion parallax, occlusion and blur in relation with the specific image on the perceived realism of the virtual window. They found that motion parallax, occlusion and blur on their own have a significant impact. The combination of motion parallax and occlusion also was significant. It remained significant in combination with the image. Furthermore occlusion, blur and the image in combination had a significant impact. The remaining combinations were not significant.

They used a simplified motion parallax (visible part moves but objects in the visible part don't move relative to each other), which shows that even such a simple motion parallax yields good results. They noted however that scenes with foreground objects likely won't work as well with the simplified version, because we expect foreground objects to cover larger distances than background objects in the same amount of time. In the simplified version both move the same distance over the same amount of time. But scenes with no foreground objects (e.g. mountains) should work very well.

This was of particular importance, because in 2006 it wasn't feasible to have full motion parallax in real-time. The simplified version however was possible in real-time. In addition they discussed that lighting conditions might have an important effect and that framing is helpful in providing depth cues.

Therefore at least the simplified motion parallax is needed for virtual windows to provide immersion to the viewer.

But why was full motion parallax not feasible? What is meant here is full motion parallax combined with photorealism. The hardware indeed wasn't capable of that in 2006. However full motion parallax in itself was available ever since Ivan Sutherland introduced head-tracked and head-coupled devices in the 1960s. These devices have been in use from that time on. Systems providing motion parallax were therefore available for decades. Some years after the introduction of the devices, head-tracked desktop systems [11] were available. They were sometimes called fish-tank virtual reality [12] and provided the user with a window-like look onto a computer generated virtual world. This world however was not photorealistic. In fact it was quite simplistic.

This means there are two development streams: One with virtual window solutions offering photorealism but no or only simplified motion parallax and another that offers full motion parallax but no photorealism. The symbolic "wedding" of these two streams is necessary for virtual windows that really behave like real windows. On the hardware side this wedding wasn't even possible until a few years ago.

Another recent hardware development allowed for wireless and marker free head tracking which is absolutely required for authentic virtual windows. The Kinect is one sensor

that allows both wireless and marker free head tracking (head is not wired to some device and no markers are needed).

Winscape [13] developed a virtual window system using HDTVs, a Kinect and some proprietary intermediary software. The TVs are mounted on a wall with an image frame around and show photorealistic videos from the real world with a simplified motion parallax employed.

Besada, Rodera, Bernardos, *et al.* [14] developed a Virtual Window System (VWS) functionally similar to that of Winscape. It uses a very complex filter to process the raw data coming from the Kinect. They use a focal point in the center of the head as the head location. More detailed positions like the eyes are not used. On the technical side they use the first version of the Kinect and describe the move to Microsoft libraries and Kinect v2 as future work. Kinect v2 was released in October 2014[15] but [14] was published in 2016.

The obvious latency between the scientific work and the published article shows that there might be more solutions out there, which just weren't published yet. Both Winscape and the VWS of Besada, Rodera, Bernardos, *et al.* [14] have in common that they use simplified motion parallax and photorealistic images/videos but no full motion parallax (objects visible in window move relative to each other).

There doesn't seem to be a commercial or scientific solution which uses both photorealism and full motion parallax which can only be realized by using a game engine or a very fast moving camera in the real world.

Chapter 3

Kinect

The Microsoft Kinect was used for the head tracking used in the technical solution developed for this bachelor thesis. It is a sensor with an HD camera, an Infrared (IR) sensor and a microphone. The following sections will go into detail about the data that is provided by the sensor and the Kinect SDK, the limitations of the Kinect and the reasons why the Kinect was used despite its limitations.

3.1 Provided Data

The data is split into the basic data and specific face data which uses an additional detection step.

3.1.1 Basic data

The Kinect has three sensors: an HD camera, an IR sensor and a microphone. These provide raw data which is provided by the Kinect SDK in the form of so called frames. As there are two words named "frame" being used, they have to be disambiguated. In the following text "KinectFrame" will refer to the data structures provided by the Kinect SDK while "UnityFrame" refers to the measurement unit for the game engine (e.g. x frames per second).

Each KinectFrame represents the data recorded for the currently displayed UnityFrame. There are classes for each of the data types: `ColorFrame`, `InfraredFrame` and `AudioBeamFrame`. The internal structure differs between them. For this thesis only the colour and infrared data is relevant. Therefore the `AudioBeamFrame` is not explained in more detail.

The method `CopyConvertedFrameDataToArray` is used to actually use the data stored in a `ColorFrame`. As first parameter it expects the array the data should be copied to and the second parameter is the desired colour format (e.g. RGBA or BGRA). In the one dimensional array there are four fields per pixel. With the RGBA format the green part of the 3rd pixel from the top left corner would be at the index 9.

The `InfraredFrame` functions in a similar manner. The data must be copied with the method `CopyFrameDataToArray`, which only expects the destination array. This array must be able to store 16-bit unsigned integers.

In addition to this raw data, the Kinect SDK provides processed data as well. The closest one to the raw data is the depth data. It is provided with a `DepthFrame`. Similar to the `InfraredFrame` the data must be copied with `CopyFrameDataToArray`. Again the array must be able to store 16-bit unsigned integers representing the depth in millimetres.

More high-level data is available with the `BodyFrame`. Following the established pattern the data must be copied with `GetAndRefreshBodyData`. Each item in the array is of the type `Body`. This class offers various high-level information. For the purpose of this thesis only the properties `Joints` and `IsTracked` are of interest. The first one contains a map of joint types to joint positions while the second is a boolean and describes whether the body is tracked by the Kinect.

To retrieve the head position from the map of joints, the value `Head` of the enumeration `JointType` must be given as key: `JointType.Head`. The returned value is of type `Joint` and here the property `Position` is of interest. It contains the position of the joint in the camera space and is of the type `CameraSpacePoint`. Such a point has the three properties X, Y and Z with the corresponding values.

The mentioned camera space refers to one of multiple spaces offered by the Kinect. The camera space is the processed 3D coordinate system originating from the IR sensor where each coordinate is given in metres. As described it is used for the body data. The colour space is a 2D coordinate system originating from the HD camera and the depth space is a 2D coordinate system originating from the IR sensor. Both 2D coordinate systems store the coordinates in pixels. As the IR sensor and the HD camera are located at different locations on the Kinect, the SDK offers a `CoordinateConverter` which allows for easy translation from one coordinate system into the other.

You can't translate a single colour space point into a camera space point, because it is missing the depth information and the colour and depth resolutions differ from each other so that it is not possible to find the depth for one specific colour space point. However you can convert an entire colour frame to camera space. In this case you will get corresponding camera space points for each colour space point but some of these points will have negative infinity for the x, y and z values as the problem with different resolutions remains.

3.1.2 Face data

The previously described data is part of the default data "package" and available every UnityFrame without a lot of extra work. But the Kinect also offers face data. However

it is a lot more difficult in comparison to obtain this data. Furthermore it depends on the body data.

The first difference is the setup. You need to explicitely set up a `FaceFrameSource` and specify the `FaceFrameFeatures`. For this thesis `PointsInInfraredSpace` is requested as feature. Next the reader is opened.

In the code that is executed for each UnityFrame, first the body must be obtained. In particular the trackingID of the body is required. This ID is given to the frame source and only then the frame is provided. This `FaceFrame` contains all the requested features for the face of the specified body. The property `FaceFrameResult` contains the actual data.

Infrared data is found under the property `FacePointsInInfraredSpace`. The data is organized as a map with `FacePointType` as keys and the corresponding `Point` values. The face point types `FacePointType.EyeLeft` and `FacePointType.EyeRight` are of interest for this thesis. The points contain the X and Y position of the face points in pixels in the previously mentioned depth space. These X and Y values must be combined with the corresponding depth to be able to translate it into camera space.

3.2 Limitations

First of all the Kinect is a relatively cheap consumer device which is primarily meant for gaming and some gadget-like applications. It was not developed for high-end high-demand sensor solutions. Considering these circumstances the data of the Kinect is very good and a high performance is provided.

For the virtual window application however the Kinect is definitely not ideally suited. The data is only good in a range from 0.5 to 4.5 metres from the sensor. This means that it is basically impossible to go as close to the virtual window as you can go to a real window. The Kinect simply couldn't recognize you anymore.

On top of the range comes the location of the Kinect in every possible setup. As the display "playing" the virtual window must be free from obstruction, the Kinect has to be placed next to the display. This means that up close the head is most likely not even in the field of view (FOV) of the Kinect.

Actually there are two field of views (FOVs): One for the colour image and one for the depth image. The colour FOV is 84.1x53.8 degrees which results together with a FullHD resolution (1920x1080 pixels) of the camera in about 22x20 pixels per degree. The depth FOV is 58.5x46.6 degrees which results together with a resolution of 512x424 pixels in roughly 7x7 pixels per degree.

This difference in resolution is another limitation in itself. It means that not every pixel in the colour image has a corresponding depth.

Both the colour and the depth is recorded with a frequency of 30Hz, which is usually half of the display refresh rate.

The aforementioned FOV results in yet another limitation. Wei, Lee, Qiao, *et al.* [16] found that the Kinect is good for frontal views and back views but not really good for side views. The back view is only relevant if the Kinect is placed opposite to the display. This would however require additional number conversions to convert the Kinect-based positions into positions that originate from the display. Bad side view capability is a problem though. It basically means that people approaching the window from the side are hardly recognized by the Kinect.

These limitations are not trivial but they represent the current state of the Kinect. There was a major step between the first Kinect version and the second. A third version or another device could therefore get rid of most of these limitations. Some limitations can already be overcome by using multiple Kinect devices and combining their data.

3.3 Why is it used?

The described limitations have a significant impact on the ability to develop fully immersive virtual windows. However in light of future hardware improvements these limitations are hopefully only temporary. Therefore this thesis uses the Kinect despite its limitations. Negative feedback regarding the immersion (if it can be traced back to the Kinect) is relativized by aforementioned prospective hardware improvements. Good feedback is an indicator that in future the results could become even better if more powerful hardware allows for more immersive virtual windows.

In addition the Kinect does provide some real benefits that outweigh the limitations. It doesn't require additional markers or hardware installation beyond the sensor itself and a computer which receives the data. The setup of a Kinect-powered experiment or virtual window solution is therefore easy and fast. The lack of markers is not only good for the people building the experiment, it is also a strong advantage for the viewers of a virtual window, because they need no extra equipment.

The Kinect SDK is another strong point as it offers high-level body data which saves a lot of work that would otherwise be spent on reinventing the wheel aka developing an algorithm that takes the raw data and calculates the body data. Such a self developed algorithm would be error prone, most likely not as performant as Kinect's solution and hard to maintain over a longer period of time.

Another reason for the Kinect is its status as consumer good. If virtual windows can be realized using only technology that is readily available for everyone, they can be used

by everyone. There might be products out there that offer fewer limitations than the Kinect but they are most likely not as affordable. A scientific research using such an elite product might provide interesting academic insights but would be mostly useless for everyday life.

Chapter 4

Own work

The upcoming sections form the contribution of this thesis. At the beginning the factors that influence the success of a virtual window solution are analyzed and their advantages as well as disadvantages discussed. After this more theoretic section the specific implementation is described. That part is split into three sections. The first section describes the general design of the implementation while the next two describe the specifics of the two approaches that were pursued. The first approach makes use of OpenCV[1]/EmguCV[2] while the second utilizes the body data provided by the Kinect SDK. Next are the sections about the performed study, followed by a discussion of the results.

4.1 Factors

There are various factors that can have an impact on the immersiveness of a virtual window. Each can have a beneficial or detrimental effect on the immersion. This is not a binary (and easy) choice however. A factor could have a very beneficial impact on the immersion but might be very costly to achieve.

The most obvious one is motion parallax aka "the content of the window adapts to the viewer's position". As described by IJsselsteijn, Oosting, Vogels, *et al.* [10] there are three modes that can be differentiated:

- no motion parallax: default for monitors
- simple motion parallax: content of window changes but you can't look for example around a tree by moving left or right
- full motion parallax: behaviour of a real window

It is pretty clear that a virtual window system needs at least a simple motion parallax to be immersive. Otherwise it is limited to exactly one spot from which it looks good.

[1] `http://opencv.org/`
[2] `http://www.emgu.com/wiki/index.php/Main_Page`

The interesting part is whether the difference between simple and full motion parallax is significant enough to justify the extra work required for it.

Full motion parallax can be achieved currently by modelling the displayed world in a game engine and using an "ingame" camera to create the visual output or by using a movable camera in the real world that responds to movement almost immediately and accurately.

Both of these ways require a lot of preparation work to achieve a really good result. The virtual world must be modelled and to have a realistic looking world, AAA gaming technology must be employed. This also means that a gaming PC is required to run the virtual world in realtime together with the head tracking. The camera in the real world requires extensive hardware setup as it must be able to move in a 3D coordinate system (left-right, forward-backward, up-down). In addition the camera must get the movement commands from the head tracking device and if both are at completely different locations, some kind of normalization between both positional systems must occur.

Long story short: It is very expensive in time effort if you want a realistic and immersive experience.

Simple motion parallax is very easy to realize in contrast. All you need is a high-resolution (preferably 4K) video stream, the always required head tracking device, a display area (showing only a part of the stream) and a game engine to move a camera in front of the video texture.

The example of motion parallax shows that it is often difficult to decide what is the "best" solution. The key to understand, if full motion parallax really offers a significant improvement, is the content of the displayed world. This leads to the next factor: The existence or non-existence of objects in the foreground.

We are used to look around close objects that occlude a part of the world. If a tree blocks part of a sign in the background we can move left or right to see the complete sign. With simple motion parallax this is impossible. Let's say the tree behaves as expected when moving ("moves" as fast as it would be in the real world). In that case by moving left or right the tree will have moved accordingly but the sign is still blocked as it moved in the same speed and covered the same distance. This will result most likely in a severe drop of immersion.

The result would be different however if there is no such foreground object, be it a tree or something else. If only some distant mountains with some trees on them are visible, the lack of full motion parallax should be hardly recognizable. The trees are so far in the distance that even in the real world a small movement to the left or right side wouldn't have an impact on the position of the trees relative to the mountains.

These two factors combined - considering the amount of time and work for full motion parallax - lead to the conclusion that simple motion parallax combined with no fore-

ground objects should yield good results with a reasonable amount of work involved.

Another factor is stereoscopic 3D. Multiple solutions are available. The biggest difference is between active 3D and passive 3D. Active 3D uses so called shutter glasses which actively close one eyeside per frame. The opposite is passive 3D using polarized surfaces and passive glasses. This technique is used in cinemas but is also available for private usage. Since both techniques require glasses and some special hardware, whatever effect the glasses have, it effects both techniques in the same way and therefore the techniques are merged with respect to this analyzation.

Stereoscopic 3D (S3D) can be off or on and poses three questions:

- Does stereoscopic 3D have a significant impact?
- If the impact is significant, is it positive or negative?
- Do viewers with active stereoscopic 3D notice a difference between simple and full motion parallax if there are no foreground objects?

It seems to be easy what the answers for these questions are but is it? S3D offers an experience closer to the real world by providing a depth cue that otherwise misses. It should therefore have a positive impact on immersion. The current technology makes this less straightforward however. Active glasses must be very close to the emitter to receive the correct signals. If they do not the closing mechanism can produce wrong results which destroys the 3D experience. In that case S3D should result in a worse position immersion-wise compared to no S3D.

Assuming there are no technical issues the interesting question remains if viewers can differentiate between simple motion parallax and full motion parallax. In case of foreground objects like a tree this is easy to answer and therefore not interesting. Without these foreground objects however it is questionable if S3D has any impact at all. Technically simple motion parallax comes with a uniform depth shared by every object visible. There might be ways to record the depth while at the same time only providing simple motion parallax but these are outside the scope of this thesis. The uniform depth means that the S3D is mostly robbed of its strength: providing depth. Depth is still available but it is the same for every object. Therefore the difference between simple and full motion parallax becomes effectively a difference between a uniform depth and individual depth for each object.

This side effect could lead viewers to recognize a difference between the scenes representing simple and full motion parallax in S3D but not because they noticed a difference in the way objects move.

All in all these are the factors used in this thesis to create different scenes and measure their significance and impact. But more on that in the study sections.

4.2 Implementation

The term implementation refers to all the custom things made for this bachelor thesis. They are essentially split into code and a 3D world. The final state of both the code and the 3D world is the result of a long process of iterative improvements and changes. At the beginning the 3D world was nothing more than a few generic cubes and spheres to achieve the proof of concept. The code was designed from the beginning to be modular to minimize the required changes and make maintenance easy. A general pipeline was designed which split the code into various classes – each responsible for a particular thing in the pipeline.

The pipeline starts with retrieving the data (the specifics are described in section 3.1) from the Kinect sensor by using the Kinect SDK API. This data is processed to determine the head location in the virtual world and finally the camera in the world is moved to the calculated position. Specifically the `*SourceManager` classes are responsible for retrieving the data and providing the data to the remaining classes via public API. They run at the beginning of each frame before all other classes. The `CameraMover` runs afterwards and is responsible for moving the camera. It asks the `HeadLocator` for the specific head location and converts that location into Unity world coordinates that can be used to modify the transform of the camera. `Vector3.Lerp` is used to prevent a jumping camera by gradually moving the camera from the previous location to the new location within a relatively short period of time. All the logic for calculating the head position is encapsulated in the `HeadLocator` which makes it relatively easy to switch from one calculation to another without impacting the rest of the code.

This basic structure was implemented early on but it was continually improved and many bugs were squashed. Especially undocumented behaviour from the Kinect SDK took some time to discover and properly deal with. For example when converting a colour frame to the camera space not all points from the colour frame have a corresponding depth point. The colour space points without depth point were converted into a camera space point with `-INF` being the value of the three axes values `x, y` and `z`. If the head location matched one of these invalid points the camera behaviour was erratic. The solution was to use a guard if clause that only proceeds with the camera movement if the new position is actually a valid one.

Later on in the development the virtual world was massively improved by using the "Island in the Ocean" asset[3] from the Unity asset store which isn't available anymore due to unknown reasons. But that asset wasn't the last one used to improve the 3D virtual world. With the help of the Unity Standard Assets the trees and the water were replaced with more realistic ones. The new trees are palm trees using SpeedTree technology while the new water is moving. The water is also capable of reflection and refraction which result in a better visual quality but are incompatible with asymmetric camera frustums

[3] `https://www.assetstore.unity3d.com/en/#!/content/36393`

which are required for the virtual window. The "GameJamMenuTemplate"[4] was used to create a basic menu and "UniStorm"[5] is used for a realistic sun movement.

On a more technical side the asset "Active Stereoscopic 3D for Unity"[6] is used to allow for stereoscopic 3D in Unity.

The `HeadLocator` was mentioned but without details about how the head location is actually determined. This is where the two approaches come into play. At first the OpenCV/EmguCV approach was followed but later rejected to use the Kinect SDK approach.

4.3 OpenCV/EmguCV approach

The Kinect SDK provides high-level body data out of the house. Despite this an effort was made to use the Viola-Jones face detection algorithm. The major reason for using the Viola-Jones algorithm is its widespread implementation and availability. Furthermore the images of any camera would suffice for the X and Y head position. Combined with some other depth measuring device, the Virtual Window System (VWS) wouldn't require a Kinect anymore. That would in effect also make the VWS platform-independent and it could be run for example on Linux. Basically the idea was to use the Kinect only as sensor and don't use any high-level data from the Kinect SDK.

As Unity doesn't provide an implementation of the Viola-Jones algorithm, it was necessary to find a library that does. OpenCV was quickly found as it is THE image processing library in C++ and used by countless applications. However the binary libraries cannot be called from Unity since it doesn't support C++ directly. Therefore a wrapper was required and EmguCV was found. It is a C# wrapper for OpenCV and was exactly what was needed. The provided binaries didn't work however. They required some part of .NET that was incompatible with Unity. However a Unity-compatible plugin was advertized on their site. But it was hidden behind a very high paywall. Luckily the source code of EmguCV is free and the required code to make it compatible with Unity is available too. Therefore EmguCV and the wrapper code for OpenCV was compiled with the necessary compile switches and that eventually allowed for testing the Viola-Jones algorithm in Unity.

Viola-Jones uses so called cascades for the face detection. By default Haar cascades are used. These provide a high resolution but are significantly slower than LBP cascades. LBP cascades are faster and still provide a good quality. The existance of LBP cascades wasn't known until later in the development.

[4] `https://www.assetstore.unity3d.com/en/#!/content/40465`
[5] `https://www.assetstore.unity3d.com/en/#!/content/2714`
[6] `https://www.assetstore.unity3d.com/en/#!/content/3367`

Once testing could begin two results were observed. First that it basically worked which proved the concept and showed that the pipeline was indeed working. The second result however was a very poor performance which made it unusable. A switch to LBP cascades was made but didn't make a noticable difference. All the available factors offered by the EmguCV API for the algorithm were tweaked to find out if any of them improved the result but none of them did in a significant enough way to make it usable. The logged frames per second (FPS) were between 6 and 8 on average which clearly shows the bad performance of the implementation.

That said many videos on the internet showed the Viola-Jones algorithm working in realtime with adequate performance. It wasn't clear however what caused the difference in performance. It might have been a local problem with the custom compiled library or something else completely. But a full analysis including source code analysis of the various implementations of the algorithm and dedicated benchmarking was out of scope for this bachelor thesis. Therefore the decision was made to abandon this approach and continue with the Kinect SDK approach.

4.4 Kinect SDK approach

The Kinect SDK approach has its name from the heavy usage of the Kinect SDK. As described in section 3.1 the Kinect SDK provides a lot of data. One part of that is the body data which is easy to use and contains everything to get you started after a short time. In particular the head position is useful. In addition the SDK provides the ability to track the face. The face detection and tracking doesn't use Viola-Jones and makes use of the body data. It takes the head position and only looks for a face close to the head location and thereby limiting the amount of time necessary for finding a face. In contrast to all the other data provided by the SDK, one must specify what exactly of the face should be tracked. The limitation of tracked items is another performance optimization for the calculation heavy face detection. Especially the eye locations are interesting for the virtual window usecase. They are more specific than a general head position and could offer a subtle but noticeable improvement for the viewing experience. However the eyes are not always visible and especially when viewing from the side it could become very difficult to keep tracking the face. In addition the face data suffers from the dependence on the body data. This means that all the shortcomings of the Kinect described in section 3.2 also apply here.

The performance is an important factor. The previous approach was abandoned due to the low performance. This approach didn't have any performance problems. From the start 60 FPS were reached. The good performance led to the decision to conduct a small study to evaluate the created virtual window system.

It is a possibility to use the face data when available and to fallback to the body

data when no face is detected. This fallback plan was implemented and it somewhat worked but especially in the fringe areas of the Kinect detection area the resulting camera movement was worse compared to only using the body data. A better and more intelligent way to combine these two data sources might exist but wasn't found in time. Instead of spending many more hours trying to fix the problems of the fallback solution with potentially no rewarding outcome, the face data was ignored and only the body data was used for the study.

That said quite a lot of time and effort went into fixing problems with Kinect detection. With the current technology and the available hardware the virtual window could only work for one person at a time. Kinect does detect multiple bodies though if there are multiple people in the room and sometimes if there are not. This posed the question which body should be used for the head tracking? At first a very simple solution was found by always using the first found body. But the order of the bodies in the body array can shift if new bodies are detected or old lost. This initial solution was therefore inadequate and made it impossible to use the virtual window with more than one person in the room. The final solution – a more sophisticated way surely exists but there was no time for more – requires that only one body is detected at the beginning. Afterwards more people can come into the detection area and leave it without impacting the head tracking. As long as the original detected body remains tracked, no other person has any impact on the tracking. Once the original detected body is lost for tracking, the first body detected (first in the body array) will be used from that point on. This was realized by saving the `bodyID` of the tracked body in a variable. Each tracked body has a unique `bodyID` which makes it easy to only use the body data from the tracked body. If the body had to switch, because the old was no longer tracked, the saved `bodyID` wasn't found anymore either. In this case the first body in the array was taken and its `bodyID` saved.

There were a bunch of other problems with the Kinect detection. One of them resulted in a camera that moved to (0,0,0) everytime the tracked body left the detection area. As a fix the camera was prevented to move anywhere once the person left the accurate detection range (which ranges from 0.5 to 4.5 metres distance from the Kinect). In addition a fix was implemented to prevent the camera from moving if the last position is too far away from the new position. In combination these fixes broke intended behaviour. If a person left the detection area on one side and reentered on another, the camera didn't move until that person went back to the "last known position". It took a while to find out that the (0,0,0) problem was actually caused by the code developed for this bachelor thesis. At some point the head location was initialized with (0,0,0) and if no body was found this value was never overridden. After fixing this problem it was possible to remove the 4.5 metres "fix". The distance between the old and new head location remains important though.

These fixes however can only go so far. They cannot overcome the already described

inherent limitations of the Kinect. Furthermore they are very basic. There are virtual window solutions available that use the Kinect and use very sophisticated means of processing the raw data. This results in much more flawless positional data. But their code isn't open and sometimes Unity isn't used at all which makes it very complex to adapt those solutions.

4.5 Study

As described in section 4.1 there are two major factors (Stereoscopic 3D (S3D) on/off, full/simple motion parallax) that could have a significant impact on how viewers perceive the virtual window. A study was designed to find out if these factors indeed have a significant impact on the results. In addition the study should find out which combination achieves the best results.

4.5.1 Participants

A total of 20 people (3 female and 17 male, ages 18–31, M = 22.9) participated in the study. The participants were students or members of the department of informatics at the University of Hamburg. The student participants obtained class credit. All but one of the participants had normal or corrected-to-normal vision. Seven participants wore glasses during the experiment. The one person with an eye vision not normal had a strong eye dominance. This presented a problem in the 3D scenes as no 3D effect was recognized. Therefore this dataset was removed from the evaluation. 8 of the participants had previously participated in studies involving augmented reality. A total of 14 participants had experience[7] with S3D before (e.g. in the cinema). The total time per participant varied a bit but it was roughly 10 to 15 minutes.

4.5.2 Material

The study was performed in one of the laboratory rooms, which was sealed off for the duration of the study. The participants wore 3D shutter glasses for all scenes – including the 2D scenes. These glasses belonged to the Optoma GT1080 beamer used for the presentation. The beamer was set to a 1920x1080 pixels resolution. The specific positioning can be seen in the Figure 4.1. The beamer was rotated by 90°, because it lacked horizontal keystone correction. Therefore it would have projected a trapeze

[7]Question: "Do you have experience with 3D stereoscopic display (cinema, games etc.)?"
Answer options: scale from 1 to 5, 1 being yes, 5 being no
Experience included answers 1 and 2

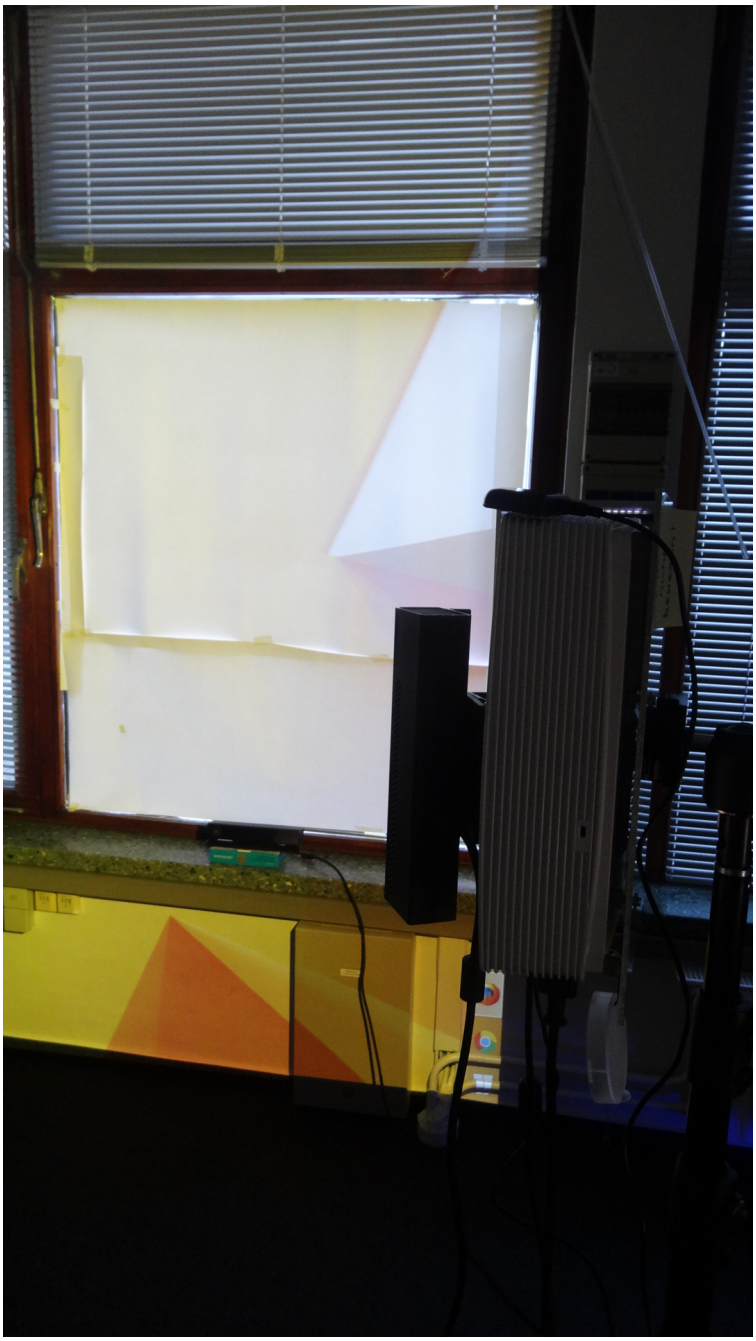

Figure 4.1: The beamer is rotated by 90° to offset the missing horizontal keystone correction. The Kinect mounted to the beamer isn't used in the study. In the background the used Kinect can be seen below the window, which is covered by paper. On the beamer lies the IR emitter for the 3D glasses which is connected to the beamer for synchronization.

onto the projection area if positioned at the same location and not rotated by these 90°. Without rotation the only place without trapeze projection would be on a line vertical to the window. But that wasn't feasible, because the participants would not have been able to move between the beamer and the window and/or watch the window unobstructed by the beamer. The necessary rotation brought additional problems though, because the 3D projection assumed left/right positions of the images. These left/right positions however translated to bottom/top positions for the viewer. Hence the shutter glasses were not able to correctly show the image for the left/right eye respectively. The "solution" was to tell every participant to rotate the head the same 90° to view the 3D correctly.

The PC used for the rendering, browsing and logging was running Windows 8 Pro 64bit and used the Intel i7 4790K with 4.00 GHz as processor. The computer had 16GB RAM and used the Nvidia Quadro K5200 as graphics processor. Unity3D was used in the version 5.3.2p3. The Microsoft Kinect was used for head-tracking purposes.

The virtual environment used for the study was created and rendered in the Unity3D engine. It was built using assets from the Unity asset store and standard assets provided by Unity itself. The detailed breakdown can be found in section 4.2. A realistic sun behaviour was implemented using Unistorm. The sun is synchronized with the local real world time which resulted in different shadow positions and lighting amount between participants. Since the sun moved at real world speed a difference could not be recognized within the 15 minutes one participant viewed at the virtual environment. The figures 4.2 and 4.4 show the morning version of the scenes. Figures 4.3 and 4.5 show the 5pm version of the scenes. The difference should be noticeable.

The participants were free to move around and look at the virtual environment in the way they wanted. This included the possibility to move too close for the Kinect to detect any head. Figure 4.6 shows how the virtual world looked through the virtual window.

The virtual window was placed in a real window for immersion purposes. This decision had a downside as well. The external sun light was very strong in the days of the study. All the real windows were darkened as best as possible with the available materials. But as you can see in Figure 4.6 the light is still quite strong. In addition the virtual window itself has therefore backlight which doesn't improve the visiblity of the virtual environment.

4.5.3 Methods

Before the study could start it was necessary to come up with a way to test a Virtual Window System (VWS). IJsselsteijn, Oosting, Vogels, *et al.* [10] asked participants to rate the "see-through experience" on a scale from Weak (0) to Strong (5). They could point to any location on the scale as the questionnaire was on paper. The actual value

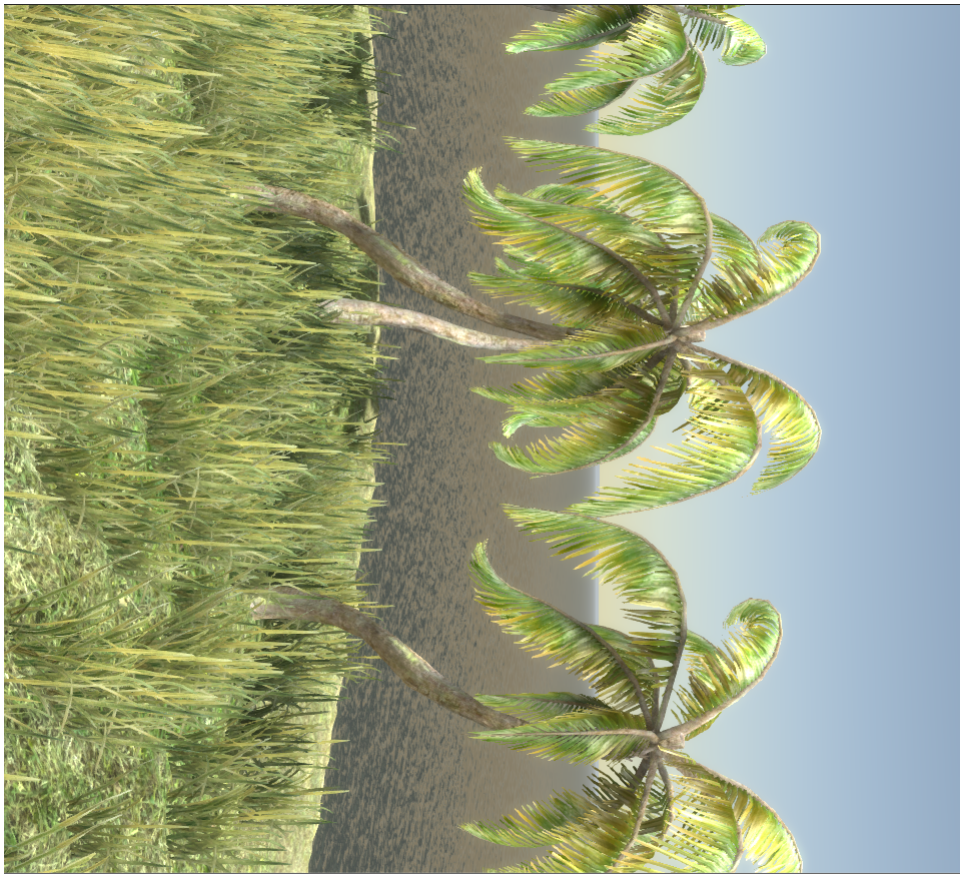

Figure 4.2: Full motion parallax scene as delivered to the beamer. The scene is shown in the 10am version.

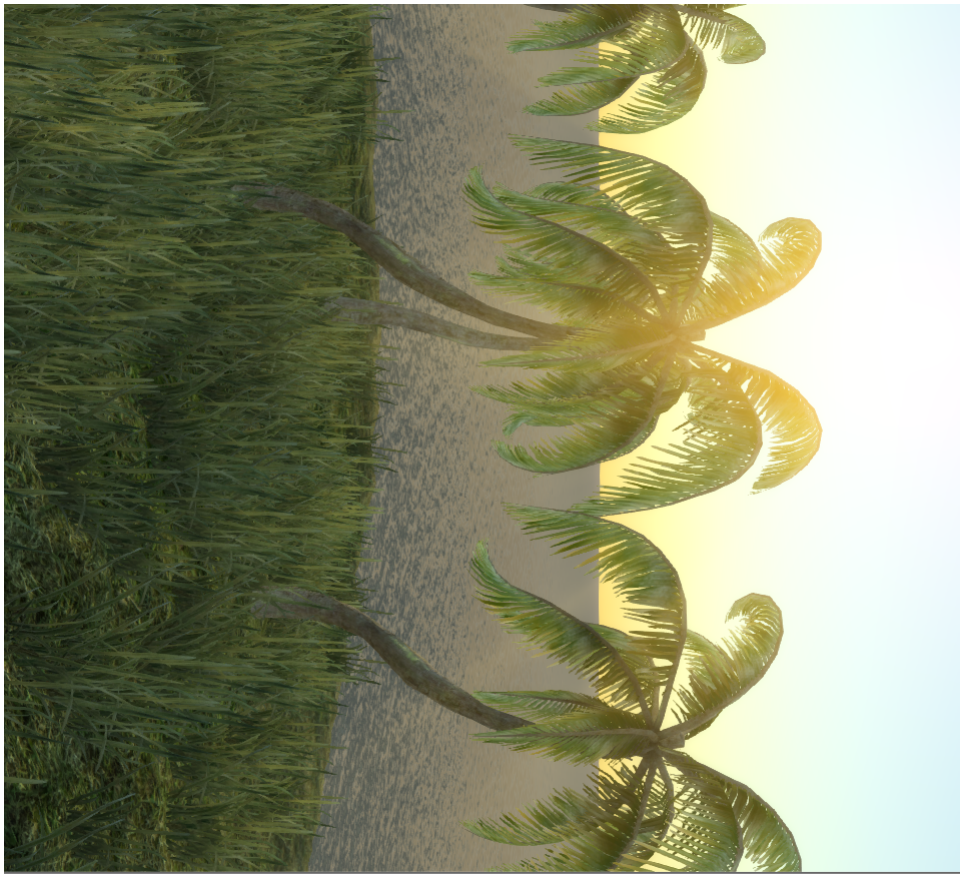

Figure 4.3: Full motion parallax scene as delivered to the beamer. The scene is shown in the 5pm version.

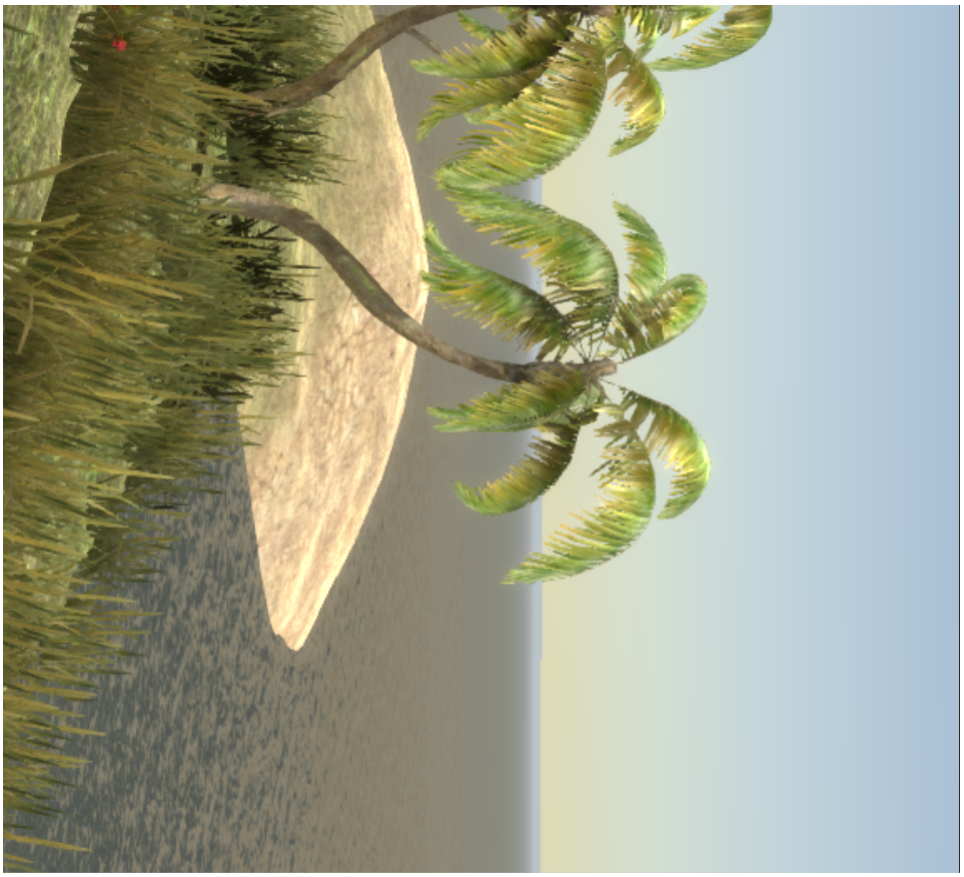

Figure 4.4: Simple motion parallax scene as delivered to the beamer. The scene is shown in the 10am version.

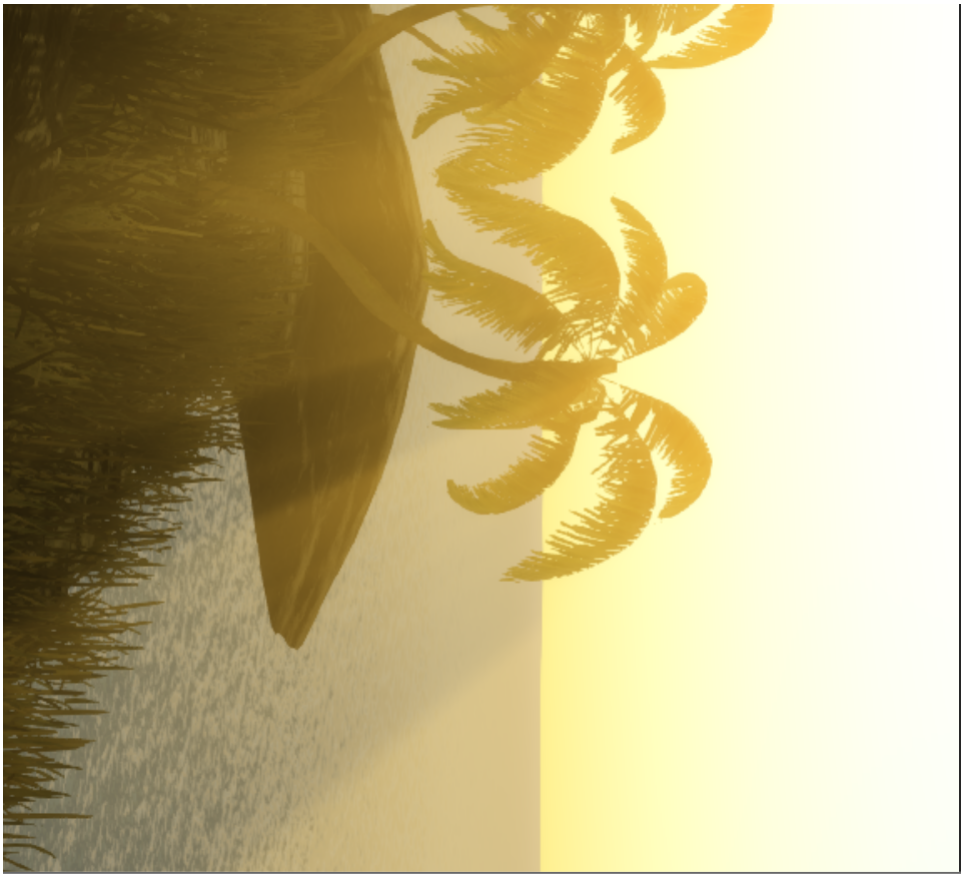

Figure 4.5: Simple motion parallax scene as delivered to the beamer. The scene is shown in the 5pm version.

Figure 4.6: The 2D scene with full motion parallax is shown. In addition the lighting issues caused by the real world sun are visible.

on the scale was measured with a ruler after the experiment. In this study digital questionnaires were used so that the measurement was easier and more uniform. The question about the see-through experience was adapted into this one: "When you think back to the experience, do you think of the virtual environment more as images that you saw or more as a place you could visit (by opening the window and going through it)?" This question essentially asks if the virtual environment is a real world right behind the window. In addition two more questions/tasks were added to judge all components of the VWS. The first one wants the participant to specify how natural the window interaction was[8] while the second asks how real the virtual environment seemed[9]. The difference between the "see-through experience" and the "realism of virtual environment" question is their focus. The first focuses on the overall experience and basically asks how immersed any participant was. The second however asks how real the virtual environment is. A good example to illustrate the difference is the Golden Gate Bridge. If you have a window and play a video of the bridge behind it on a large 4K display it is definitely a real environment but it can still feel more like a flat image/movie than a real window with a real environment behind it. The questions are based on the Slater-Usoh-Steed presence questionnaire[17].

There are two conditions that can have an important impact on the outcome of any of

[8]Please rate your feeling of looking through a window, on a scale of 1 to 7, where 7 represents your normal experience of viewing through a window.

[9]To what extent were there times during the experience when the virtual environment seemed real for you?

the above mentioned questions. They were already mentioned in the section 4.1 and are the existance of stereoscopic 3D and the used version of motion parallax (full or simple as described in [10]). This results in a total of four possible combinations of these conditions.

- S3D off, Full Motion Parallax
- S3D off, Simple Motion Parallax
- S3D on, Full Motion Parallax
- S3D on, Simple Motion Parallax

Since there could be strong interpersonal differences in the perceived experience of all these four scenes, the decision was made to use a within-subjects design for the study. This means that all participants viewed all scenes. The order of the scenes was not the same for every participant though. Every second participant started with 3D instead of 2D and there were no cases in which two 3D or 2D scenes were back-to-back. In addition to this even 50% split the exact scene was randomized as well. Since there are two 3D and two 2D scenes to choose from, a total of eight different combinations were possible.

- 2D/FMP (Full Motion Parallax), 3D/FMP, 2D/SMP (Simple Motion Parallax), 3D/SMP
- 2D/SMP, 3D/FMP, 2D/FMP, 3D/SMP
- 2D/FMP, 3D/SMP, 2D/SMP, 3D/FMP
- 2D/SMP, 3D/SMP, 2D/FMP, 3D/FMP
- 3D/FMP, 2D/FMP, 3D/SMP, 2D/SMP
- 3D/SMP, 2D/FMP, 3D/FMP, 2D/SMP
- 3D/FMP, 2D/SMP, 3D/SMP, 2D/FMP
- 3D/SMP, 2D/SMP, 3D/FMP, 2D/FMP

Before the experiment, participants filled out an informed consent form and the first part of the demographics questionnaire. This first part contained general demographic data like age, gender, course of study and more specific data like vision correction, known eye disorders, previous experience with augmented reality, previous experience with stereoscopic 3D and a question about the type of gamer the participant is[10]. In addition a subject ID was entered by the experiment leader to link the demographic

[10]The available options were not really satisfactory which made it difficult for most to answer this question properly.

questionnaire responses to the responses of the main questionnaire. After the participants finished the first section of the questionnaire they were given the 3D glasses together with instructions to move around freely.

These instructions are obviously not as strict and repeatable as the ones used by IJsselsteijn, Oosting, Vogels, *et al.* [10] but they offer a much greater potential to find out how natural the window interaction is. If the instructions already limit the potential interaction to a minimum the resulting experience cannot be natural in any way. On the other hand the given instructions could surely be improved and become more repeatable without loosing the ability to test natural window interaction.

After a reasonable amount of time the next scene was loaded via an integrated menu. The exact amount of time varied between the participants and was decided on the spot by the experiment leader. The main condition for moving on was the behaviour of the participant. As soon as the participant seemed to have explored all angles or started to act in a repetitive way the next was loaded. Alternatively if none of these cases happened the next scene was loaded when it felt appropriately. This is another aspect which is hardly repeatable in the same way. But yet again the goal was to strike a balance between repeatability and the ability for each participant to fully experience each scene. Since the used strategies differed from participant to participant it would have been difficult to apply a uniform way of testing it. A better way surely exists and potentially could be found by looking at the used strategies to find a better spot on the scale between free form and repeatability.

At the end of all the four scenes the participants filled out the main questionnaire with the aforementioned questions. The questions were repeated for each of the four scenes and divided into sections. In addition a subject ID was entered by the experiment leader. After the participants finished this questionnaire they moved to the second part of the demographics questionnaire which asked for the level of attention, the used strategy, if participants would use a virtual window in their office and optionally for additional observations and comments.

The participants were not informed in the beginning that they would have to answer these questions for each of the scenes and they didn't know the amount of scenes to be shown or what the exact differences between the scenes were. This could have led to more uniform answers, because the participants didn't know that they had to give differentiated responses for each of the scenes and potentially didn't remember the specific differences of all scenes.

4.5.4 Hypotheses

It is significantly cheaper to do simple motion parallax than full motion parallax. Therefore it would be beneficial if no significant difference in the answers to the study questions

could be found between simple and full motion parallax. In case of stereoscopic 3D it would be beneficial if it doesn't provide a significant improvement. However the belief is that both full motion parallax and S3D do have a significant impact. For S3D it is uncertain whether that impact will be positive or negative. Hence the following hypotheses are tested.

H1 Full motion parallax has a significant impact

H2 Full motion parallax is better than simple motion parallax

H3 Stereoscopic 3D has a significant impact

4.6 Results

The results were analyzed with paired t tests at the 5% significance level. Before the actual analysis could start it was necessary to reorder the response data from the study questionnaire to match it with the order of scenes for each participant. In addition it was brought into R compatible format so that it was relatively trivial to perform the actual tests and calculate the mean values. For reference: The likert scales used to obtain the raw values aggregated into these mean values range from 1 to 7 where 7 is the best value and 1 the worst.

The independent variables were the usage of stereoscopic 3D (yes/no) and the used version of motion parallax (full/simple). The dependent variables were the natural window behaviour, the realism of the virtual environment and the perceived immersion (real world directly behind window). For the testing one of the independent variables was kept the same while the other was manipulated. This resulted in four different tests per question.

- use full motion parallax, compare between 2D and 3D
- use simple motion parallax, compare between 2D and 3D
- use 2D, compare between full and simple motion parallax
- use 3D, compare between full and simple motion parallax

In total 12 tests were performed for this study.

4.6.1 Natural window behaviour

The first question was about the natural window behaviour. The mean values for the four combinations are shown in Table 4.1. SMP delivers the worse results compared to the FMP result with the same stereoscopic 3D status (used/not used). Both 2D values

Combination	Mean
2D & Full motion parallax (FMP)	3.68
2D & Simple motion parallax (SMP)	3.37
3D & FMP	3.26
3D & SMP	2.74

Table 4.1: Means of the natural window behaviour. 2D and Full motion parallax deliver the best result. 3D and Simple motion parallax give the worst result. Both 2D results are better than the 3D results and both Full motion parallax results are better than their respective Simple motion parallax counterparts.

are better than the 3D values. But it is also obvious that on average the results lie between 2 and 4 on a scale from 1 to 7 which indicates that overall the used virtual window system should not be used in production. These values are not statistically relevant though. They could lie within the statistical margin of error unless there is a significant difference which leads to the aforementioned t tests.

The first test (for reference look at the beginning of this section) revealed that there is no significant difference between 2D and 3D while using FMP ($t = 1.0535$, $df = 18$, $p = 0.3061$). This result is already symbolized by the fact that both FMP mean values differ by less than 0.5 which clearly is not a significant difference. A different set of participants could have reversed this outcome.

No significant differences were found by the second test ($t = 2.0513$, $df = 18$, $p = 0.05508$) which means that the hypothesis H3 cannot be proven for natural window behaviour. Therefore the null hypothesis is still valid and no significant impact of stereoscopic 3D on the natural window behaviour could be found.

The third ($t = 0.9$, $df = 18$, $p = 0.38$) and fourth ($t = 1.8824$, $df = 18$, $p = 0.07605$) test revealed no significant differences between SMP and FMP (both using 2D and 3D). Hence the hypotheses H1 and H2 cannot be proven and their respective null hypotheses remain valid. Since no significant difference between SMP and FMP was found H2 cannot be proven regardless (even if mean values indicate better results for FMP compared to SMP).

4.6.2 Realism of Virtual Environment

The second question was about the realism of the virtual environment. The mean values for the four combinations are shown in Table 4.2. Full motion parallax gives the best results with almost no difference between 2D and 3D. Simple motion parallax is worse than full motion parallax in any case. The 3D values are generally worse than the respective 2D values but only for simple motion parallax there is a noticeable difference. The values are between 2 and 3 on a scale from 1 to 7.

Combination	Mean
2D & FMP	3.05
2D & SMP	2.63
3D & FMP	3.00
3D & SMP	2.26

Table 4.2: Means for the realism of the virtual environment. 2D and Full motion parallax deliver the best result. 3D and Simple motion parallax give the worst result. Both Full motion parallax results are better than all Simple motion parallax results.

Combination	Mean
2D & FMP	3.47
2D & SMP	3.00
3D & FMP	3.26
3D & SMP	2.89

Table 4.3: Means for the visitability of the virtual environment. 2D and Full motion parallax deliver the best result. 3D and Simple motion parallax give the worst result. Both Full motion parallax results are better than all Simple motion parallax results.

The first ($t = 0.175$, $df = 18$, $p = 0.8631$) and second ($t = 1.3787$, $df = 18$, $p = 0.1849$) test revealed no significant difference between 2D and 3D (both using full and simple motion parallax). This means that H3 cannot be proven and the null hypothesis remains valid.

The third test revealed no significant difference between full and simple motion parallax using 2D ($t = 1.4069$, $df = 18$, $p = 0.1765$) which means that neither H1 nor H2 can be proven and the null hypotheses remain valid.

The fourth test however did reveal a significant difference between full and simple motion parallax using 3D ($t = 2.4208$, $df = 18$, $p = 0.02628$) which proves H1 and the null hypothesis can be rejected.

4.6.3 Visitability of the Virtual Environment

The third question was about the visitability of the virtual environment. The mean values for the four combinations are shown in Table 4.3. FMP gives the best results. SMP is worse than FMP in any case. The 3D values are worse than the respective 2D values. The values are between 2 and 4 on a scale from 1 to 7.

The first ($t = 0.5926$, $df = 18$, $p = 0.5608$) and second ($t = 0.417$, $df = 18$, $p = 0.6816$)

test revealed no significant differences between 2D and 3D (both using full and simple motion parallax). This means that H3 cannot be proven and that the null hypothesis remains valid.

The third ($t = 1.2063$, $df = 18$, $p = 0.2433$) and fourth ($t = 1.278$, $df = 18$, $p = 0.2175$) test revealed no significant differences between full and simple motion parallax (both using 2D and 3D). Hence the hypotheses H1 and H2 cannot be proven and their respective null hypotheses remain valid. Since no significant difference between SMP and FMP was found H2 cannot be proven regardless (even if mean values indicate better results for FMP compared to SMP).

4.7 Discussion

What do these results mean? This section will explore potential reasons for the previously mentioned results and what they mean.

4.7.1 Potential reasons for the results

The study revealed a significant difference of the realism of the virtual environment between full motion parallax and simple parallax when using 3D. And the full motion parallax had the better results when looking at the mean values in Table 4.2. More interesting however are all the results that did not reveal a significant difference.

Does this mean that there is no difference between 2D and 3D or full motion parallax and simple motion parallax when judging how natural the window interaction or how visitable the virtual environment is? No, it doesn't, because that wasn't confirmed either. Essentially these results have no meaning at all in statistical terms. Statistically it's like this study never happened, because all the insignificant results can just be in the margin of error and another study could get different relative results.

When looking at the results regardless of significance and the potential reasons for them, then it becomes quite obvious that the specific technical implementation of the study setup was a variable unconsidered when designing the study. The used hardware except the Kinect was not specifically chosen for the study but an environmental limitation to contend with. A better beamer and therefore an improved 3D experience could have resulted in overall better results for 3D. But it would not have turned the mean values from 2 to 4 up to 5 to 7. So there must be more to it and scapegoating the environment for poor results won't be useful either.

For once the graphical fidelity of the presented virtual environment is not of AAA quality and could be of better photorealistic quality. Such an improved quality could have a positive impact on the realism of the virtual environment. Another reason for the results

especially for the natural window interaction could be the lack of conscious knowledge about how windows actually work. Many participants who gave a rather bad rating for the natural window interaction also mentioned (in person or in the demographics questionnaire under additional observations) that they found it odd how objects grew in size when moving away from the window. But that is what happens with real windows as well. It could be that the exact growth amount is off and therefore it doesn't feel natural. But it could be something else as well. Before and after the study short videos were made and when looking on the camera display while moving around, it actually felt absolutely natural. It is up for debate what exactly is the reason for this difference but understanding it could prove beneficial for virtual window technology.

The visitability of the virtual environment or, in more simple terms, "immersion" was surely held back by the Kinect's inability to detect at very close range. Therefore it wasn't possible to go as close to the window as it is possible in the real world.

But most of these points refer to the absolute results. Better overall absolute results with comparable relative differences would have resulted in an equal amount of insignificant results – just on higher niveau. The participants didn't know in advance that they had to judge four different scenes and give differentiated answers to three questions. The answers could have looked dramatically different by informing participants about this very thing up front. In addition most participants likely didn't know which parts of the study setup were environmental limitations so they included them in the judgement. One participant with extensive previous study experience however gave more positive and differentiated answers which indicate that a better understanding about the environmental limitations was present so that these limitations were not included in the judgement.

4.7.2 Meaning of results

After the potential reasons for the results have been analyzed, it's time to look at the results themselves and what they can mean. The 2D scenes offered a better natural window interaction than the 3D scenes when using full motion parallax. Considering the disadvantage of 3D in the setup, the difference between 2D and 3D is likely to shrink in a more 3D friendly environment but 3D most likely won't be significantly better. Therefore 2D is the way to go for rendered environments. It is not surprising that 3D delivers a worse result than 2D for simple motion parallax as the missing depth was likely easily spotted. The difference is also larger than in the case of full motion parallax. In any case full motion parallax is better than simple motion parallax. But the difference is small enough in the case of 2D that the use of real world images with high resolution could overturn that result.

Looking at the realism of the virtual environment, both 2D and 3D resulted in almost identical means for full motion parallax. Considering the disadvantage of 3D in the

specific setup it leads to the conclusion that 3D is at least as good as 2D if not better for this combination. But it also shows that 3D won't be significantly better than 2D. Therefore it should be possible to use 2D which is cheaper and easier to realize without a disadvantage in the realism of the virtual environment. The difference between full and simple motion parallax on the other hand is quite noticeable and indicates that full motion parallax is the way to go for rendered environments. If using real world material however this effect could be countered by using very high resolutions and not using foreground objects. Since rendered environments and photorealism still don't align perfectly well, this will likely delay any commercial products using full motion parallax, because rendered environments are the only cost effective choice for using full motion parallax. Another interesting point is the difference between 2D and 3D when using simple motion parallax. This implies that 3D does indeed make it easier to recognize the flatness in the simple motion parallax scenes. But given the disadvantage of 3D in the setup, the difference probably won't be larger in a more 3D friendly environment. In combination with the use of real world images and videos, it is clear that 2D is the way to go even though the real world is in 3D.

When looking at the visitability of the virtual environment or the immersion the previously mentioned trends are stabilizing. Again 2D and 3D have a difference with 3D on the loosing end. With a better 3D implementation it is likely as good as 2D. Full and simple motion parallax have a noticeable difference as well with it being larger when using 2D. This indicates that full motion parallax didn't benefit that much from 3D. Otherwise the difference should be larger and not smaller.

Summarizing this 2D still trumps 3D so long as 3D requires additional hardware and generally requires more work than 2D on the part of the viewer. Full motion parallax is overall better than simple motion parallax – as expected in hypothesis H2 – but requires rendered environments which are not yet photorealistic in realtime. Therefore simple motion parallax will continue to be used in conjunction with real world images and videos where the discrepancy isn't that big (e.g. no pixelation even when using simple motion parallax).

Chapter 5

Outlook

In the last few sections the results have been presented, potential reasons for them and their direct meaning have been explored. This chapter will look into the future and explore what can be done to build upon the results presented in this thesis.

On the side of science future work should repeat the comparison done in this thesis – using full vs. simple motion parallax and 2D vs. stereoscopic 3D as the independent variables – with better technology (e.g. more photorealistic virtual environment, better sensor), more participants, less environmental limitations and potentially better measurable questions. In addition long time studies can be of interest to find out if virtual windows hold up to everyday usage outside of a conscious testing environment. It should be clear that laboratory tests – as good as they can be – aren't as useful as studies in the "wild" when it comes to measuring how natural the window interaction is. Moreover it could be explored if certain demographics have an impact on the results. Especially the gaming, Virtual Reality (VR) and 4K experience could have an impact on the perceived realism of the virtual environment. Something to consider with a more realistic and photorealistic virtual environment is the uncanney valley. If the rendered environment becomes too realistic then it will be compared to the real world and if it doesn't live up 100%, it will feel odd. The results of such future studies will be tremendously valuable for the advancement of virtual window technology as a whole.

The list of potential applications for this technology is very long. Some applications are already used and can be improved. These are mostly real world applications where a real world scene is presented. Therefore the outlook will focus on potential applications for a completely rendered environment. First the virtual environment must be massively improved beyond what was possible in this thesis. One could cooperate with professional game studios working with Unity to use some of their scenes and adopt them for virtual window usage. Once the virtual window technology is advanced enough to detect from virtually all ranges and angles, it could become a commercial product for fans that can use these virtual windows to look into their favourite worlds. Thinking even further this could be used with a live game so that you can actually see gameplay happening through your window. Once it is possible to work with other major engines as well, well known AAA games come to mind. Perhaps sometime in the future the virtual

window technology could be combined with Star Citizen[1], which already supports VR, to transform your home into your own spaceship or space station where you can look out the window into the real live game world.

This is just a glimpse of what could be accomplished with virtual window technology and full motion parallax. The bottleneck are basically the sensor capabilities and the head detection. This could be tackled by developing real world windows with included sensors, the ability to darken themselves and an integrated display layer which is transparent by default but can be turned into a projection area. That way a real world window would be merged with a virtual window and the best of both worlds could be achieved. It could come with an API that combines the sensor output from each of the sensors and calculates the head position from that, allowing up close detection from any angle. If this thing is even possible it would be important from a safety point of view that by default it works as an ordinary real world window without any differences. Only by connecting it with a power socket the additional features should come online.

The interesting questions are if all of this is a) possible, b) feasible, c) cost effective and d) if the previous questions are answered with yes, when it will be done. Future work will have to answer these questions though, because that clearly escapes the scope of this thesis.

[1] https://robertsspaceindustries.com

Bibliography

[1] C. Eisler. (Jan. 31, 2012). Kinect for windows is now available, Microsoft, [Online]. Available: `https://blogs.msdn.microsoft.com/kinectforwindows/2012/01/31/kinect-for-windows-is-now-available/` (visited on 04/27/2016) (cit. on p. 2).

[2] M. C. Finnegan and L. Z. Solomon, "Work attitudes in windowed vs. windowless environments", *The Journal of Social Psychology*, pp. 291–292, 1981 (cit. on p. 3).

[3] R. Kaplan, "The nature of the view from home: psychological benefits", *Environment and Behavior*, vol. 33, pp. 507–542, 2001 (cit. on p. 3).

[4] R. S. Ulrich, "View through a window may influence recovery from surgery", *Science*, vol. 224, pp. 420–421, 1984 (cit. on p. 3).

[5] K. A. Lassonde, C. A. Gloth, and K. Borchert, "Windowless classrooms or a virtual window world: does a creative classroom environment help or hinder attention?", *Teaching of Psychology*, vol. 39, no. 4, 2012 (cit. on p. 4).

[6] TESS: Therapeutic Environmental SolutionS. (2016), [Online]. Available: `https://www.linkedin.com/company/tess-therapeutic-environmental-solutions` (visited on 04/27/2016) (cit. on p. 4).

[7] Perrin Photographic Art. (2009). Perrin Photo Art - Virtual Window, [Online]. Available: `http://www.perrinphotoart.com/virtual-window.htm` (visited on 04/27/2016) (cit. on p. 4).

[8] Sky Factory. (2016). Sky Factory, [Online]. Available: `http://www.skyfactory.com/` (visited on 04/27/2016) (cit. on p. 4).

[9] J. Häkkilä, O. Koskenranta, M. Posti, L. Ventä-Olkkonen, and A. Colley, "Clearing the virtual window: connecting two locations with interactive public displays", in *Proceedings of the 2nd ACM International Symposium on Pervasive Displays*, 2013, pp. 85–90 (cit. on p. 4).

[10] W. IJsselsteijn, W. Oosting, I. Vogels, Y. de Kort, and E. van Loenen, "Look at or looking out: Exploring monocular cues to create a see-through experience with a virtual window", in *9th Annual International Workshop on Presence*, International Society for Presence Research, Aug. 24–26, 2006, pp. 83–92 (cit. on pp. 5, 12, 21, 27, 28).

[11] S. S. Fisher, "Viewpoint dependent imaging: an interactive stereoscopic display", Master's thesis, Massachusetts Institute of Technology, 1981 (cit. on p. 5).

[12] C. Ware, K. Arthur, and K. S. Booth, "Fish tank virtual reality", in *CHI '93 Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, 1993, pp. 37–42 (cit. on p. 5).

[13] RationalCraft.com. (2013). Winscape, [Online]. Available: `http://www.rationalcraft.com/Winscape.html` (visited on 04/27/2016) (cit. on p. 6).

[14] J. A. Besada, J. M. Rodera, A. M. Bernardos, J. Portillo, and J. R. Casar, "Design and user experience assessment of kinect-based virtual windows", *Journal of Ambient Intelligence and Smart Environments*, vol. 8, no. 2, pp. 169–187, 2016 (cit. on p. 6).

[15] Kinect for Windows Team. (Oct. 22, 2014). Big day for kinect developers, Microsoft, [Online]. Available: `https://blogs.msdn.microsoft.com/kinectforwindows/2014/10/22/big-day-for-kinect-developers/` (visited on 04/27/2016) (cit. on p. 6).

[16] T. Wei, B. Lee, Y. Qiao, A. Kitsikidis, K. Dimitropoulos, and N. Grammalidis, "Experimental study of skeleton tracking abilities from microsoft kinect non-frontal views", in *Proceedings of 3DTV-Con*, IEEE, Jul. 8–10, 2015, pp. 1–4 (cit. on p. 10).

[17] M. Usoh, E. Catena, S. Arman, and M. Slater, "Using presence questionaires in reality", *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 5, pp. 497–503, 1999 (cit. on p. 26).

Glossary

API	Abstract Programming Interface. 15, 17, 36
BGRA	Blue-Green-Red-Alpha. 7
FMP	Full motion parallax. 29–31
FOV	field of view. 9, 10
FPS	frames per second. 17
HD	High Definition. 7, 8
IR	Infrared. 7, 8
POV	point of view. 2
RGBA	Red-Green-Blue-Alpha. 7
S3D	Stereoscopic 3D. 14, 19, 29
SDK	Software Development Kit. 7, 8, 10, 12, 15–17
SMP	Simple motion parallax. 29–31
VR	Virtual Reality. 35, 36
VWS	Virtual Window System. 6, 16, 21, 26

Ich bin damit einverstanden, dass meine Arbeit in den Bestand der Bibliothek eingestellt wird.

Ort, Datum	Unterschrift

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Bachelor Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel — insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen — benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum	Unterschrift