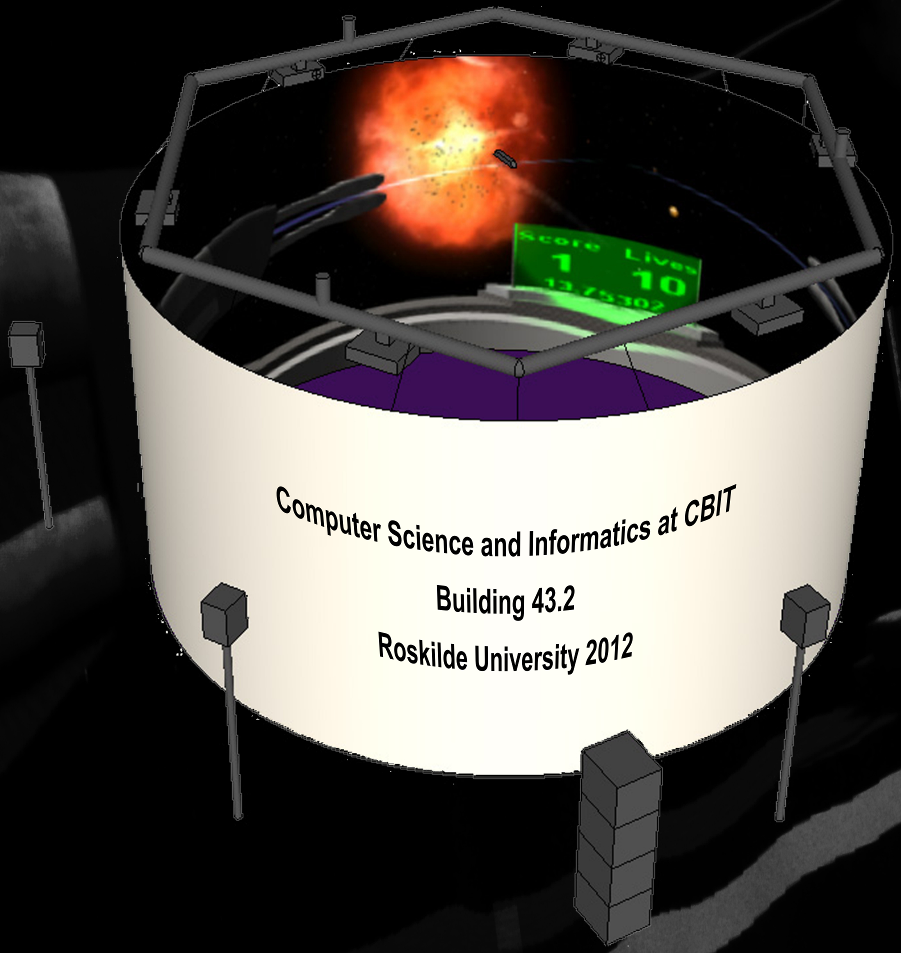
Investigating The Experience Cylinder

An Immersive 3D Game On a 360° Panoramic Display

Master Thesis. 1. February - 31. August 2012.

Supervisors: Bjørn Christensen & John P. Gallagher

Secondary supervisor: Erik Kristiansen



Computer Science and Informatics at CBIT

Building 43.2

Roskilde University 2012

Morten Brandrup (mortebr@ruc.dk)

Steffen Thorlund (thorlund@ruc.dk)

Mads Hald Jørgensen (mhaldj@ruc.dk)

Acknowledgements

We would like to thank some people involved in the process. First off we like to thank our supervisors Bjørn Christensen, John Gallagher and Erik Kristiansen as well, as a secondary supervisor, for tremendous presence and involvement when needed the most. Additionally we like to thank all of the researchers and technical staff involved in creating the Experience Cylinder for the possibility to work alongside of their development. Especially with a focus on Sisse Siggaard Jensen for being the primary advocate for our suggestion to buy Unity3D, Nikolaj Møbius for aid in Max 6, Nicolas Padfield and the technical staff at CBIT for solving hardware issues quickly, and not the least Torben Kjærgaard for presenting our project on the Experience Cylinder website (RUC 2012).

Abstract

The starting point of this project was the interactive installation called *the Experience Cylinder*. We wanted to go beyond its original intended applications in the area of interactive storytelling and consider its potential for supporting immersive experiences. Games are archetypal immersive applications that provide possibilities for exploiting the technologies available.

In this thesis, we will investigate how the Experience Cylinder can be used to support immersive game experiences.

We explore the concept of immersion, with a particular focus on games and virtual reality systems. Through the development of game prototypes we identify challenges in creating immersive game experiences that utilise the capabilities of the Experience Cylinder.

We design and implement two prototypes of interactive games. Based on our exploration of the concept of immersion we create a model with which to evaluate immersive game experiences in the Experience Cylinder. Finally we apply our model to analyse our prototypes.

Through this we demonstrate the ability of the Experience Cylinder to support immersive game experiences.

Short Table of Contents

1 Introduction 7

2 Method 11

3 Conditions 12

4 Proof of Concept 13

5 Investigation of Immersion 14

6 Building a Model of Immersion 31

7 The Experience Cylinder 36

8 Development Platform 42

9 Choosing a Case 46

10 The Development Challenges 49

11 Description of the Finished Prototypes 86

12 Analysing the prototypes through our model 88

13 Discussion 90

14 Recommendations 95

15 Literature 99

Long Table of Contents

1 Introduction 7

1.1 Problem Statement 10

2 Method 11

3 Conditions 12

4 Proof of Concept 13

5 Investigation of Immersion 14

5.1 Central Concepts Related to Immersion 14

5.1.1 Flow 15

5.1.2 Presence 16

5.1.3 Cognitive Absorption 17

5.2 Game Immersion 17

5.2.1 What is a Game? 18

5.2.2 What is a Video Game? 20

5.2.3 Four Subtypes of Immersion 22

5.2.4 Overview Table 26

5.3 Summary 27

5.4 Virtual Reality Immersion 27

5.4.1 What is Virtual Reality? 27

5.4.2 Immersion and Presence in Virtual Reality 30

6 Building a Model of Immersion 31

6.1 Construction 31

6.2 Explanation of the Model 34

6.3 Preliminary Conclusion 35

7 The Experience Cylinder 36

7.1 Introduction 36

7.2 Similar Installations 37

7.2.1 Disney Quest Example 37

7.2.2 Star Wars Example 38

7.2.3 Comparison 38

7.3 Construction and Technologies 39

7.3.1 Rigging 40

7.3.2 Display Surface 40

7.3.3 Projectors ... 40
7.3.4 Kinect ... 40
7.3.5 Computer ... 40
7.3.6 Sound ... 40
8 Development Platform ... 42
8.1 Unity3D ... 43
8.2 OpenNI ... 43
8.3 SketchUp ... 44
8.4 Max 6 ... 44
8.5 Audacity ... 45
9 Choosing a Case ... 46
9.1 Basic Case Concepts ... 47
9.1.1 Rail-Shooter ... 47
9.1.2 Base Defence ... 48
9.1.3 Virtual Sightseeing ... 48
9.2 Choice of Case ... 48
10 The Development Challenges ... 49
10.1 Displaying a 3D world on a 360° display ... 50
10.1.1 Platform Issues ... 50
10.1.2 Display Surface Limitations ... 50
10.1.3 Projectors and Placement ... 50
10.1.4 Virtual Camera in Unity3D ... 53
10.1.5 Background Summary ... 55
10.1.6 Platform ... 55
10.1.7 360° Camera ... 55
10.1.8 Virtual Optical Filter ... 60
10.1.9 Summary of Solutions ... 65
10.2 Control ... 66
10.2.1 The Microsoft Kinect ... 66
10.2.2 Kinect Driver Model ... 67
10.2.3 Background Summary ... 67
10.2.4 Tracking ... 67
10.2.5 Smoothing the Movement of Aiming Sight ... 68
10.2.6 Summary of Solutions ... 69
10.3 Directional Sound ... 69

10.3.1 Sound Setup 70
10.3.2 Sounds in Unity3D 71
10.3.3 Background Summary 71
10.3.4 Sound Setup 72
10.3.5 Directional Sound 73
10.3.6 VBAP 74
10.3.7 µ Max-Unity3D Interoperability Toolkit 76
10.3.8 netsend & -receive 77
10.3.9 Main patch and Additional Sub Patches 77
10.3.10 Editing in Audacity 79
10.3.11 Summary of Solutions 79
10.4 Creating Game Elements 79
10.4.1 Physics 79
10.4.2 The Unity3D Physics System 80
10.4.3 Scoreboard and Lives 80
10.4.4 Background Summary 81
10.4.5 Trajectory Calculations 81
10.4.6 The Meteor Game Elements 84
10.4.7 The Cannon Tower 84
10.4.8 RUC3D 84
10.4.9 Implementing Scoreboard and Lives 85
11 Description of the Finished Prototypes 86
11.1 Meteor Defence 86
11.2 A Stroll on RUC prototype 87
12 Analysing the prototypes through our model 88
13 Discussion 90
13.1 Conclusion 93
14 Recommendations 95
14.1 Display Surface 95
14.2 Projector Position 95
14.3 Stereoscopic 3D 97
14.4 Kinect Position 98
14.5 Speaker Positions 98
15 Literature 99

1 Introduction

The starting point of this project was the interactive installation called the *Experience Cylinder*. This was originally a product of interdisciplinary project collaboration between Roskilde Viking Ship Museum and researchers at Roskilde University. The goal was to provide an interactive platform for telling the story of the voyage of the Viking ship *The Sea Stallion* from Roskilde to Dublin and back.

The design of the Experience Cylinder was based on a circular space approximately six meters in diameter and three meters high functioning as a metaphor for this round trip, which is why its main feature – the display formed by the inside wall of the circular space – was cylindrically shaped. Images can be projected on the inside of the cylinder by six ceiling mounted projectors, and speakers are placed outside the cylinder to support directional sound. An infrared, depth measuring camera is mounted in the ceiling in the centre of the cylinder and can be used to track user movement within the cylinder. All these technologies are connected to a stationary computer, so other means of control, such as mouse and keyboard, are also available.

The media displayed in the cylinder respond to tracked movements of a person walking around inside the space. Using this platform the designers of the installation aimed to create an implicit narrative of the voyage which was unfolded by the user as he/she walked around inside the cylinder. A full description of the project can be found in (Andreasen, et al. 2011).

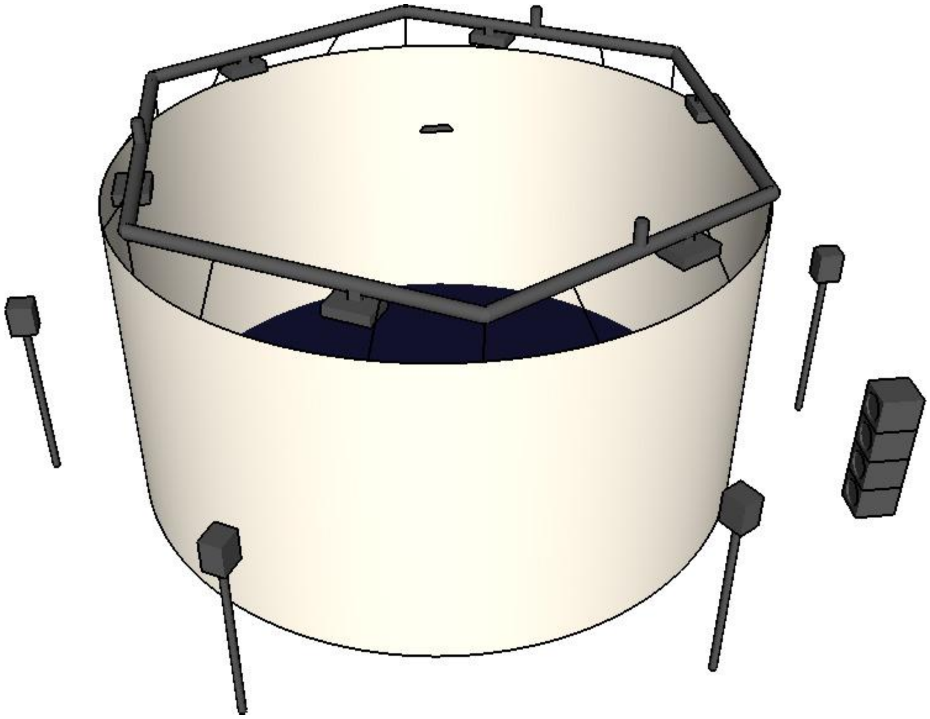


Illustration 1-1: This illustration shows a simplified 3D model of the basic hardware setup that comprises the Experience Cylinder.

The composition of technologies in the Experience Cylinder suggests that it can be categorised as an example of a larger class of systems intended for creating immersive, interactive experiences. Other well known examples are virtual reality systems and a class of system known as a CAVE. CAVE is an acronym for *Cave Automatic Virtual Environment* where projectors display a computer-generated 3D environment on the sides of a room-sized box. This enclosure shields out stimuli from the real world. The Experience Cylinder has a number of similarities to that of a CAVE, but differs from the original CAVE design in primarily two senses: first, the screen is cylindrical as opposed to the box shape of a CAVE secondly; the Experience Cylinder was originally made for presenting and navigating images and video rather than a virtual 3D environment.

The resemblance of the Experience Cylinder to CAVEs led us to go beyond its original intended applications in the area of interactive storytelling and consider its potential for supporting immersive experiences. Thus, in this project we wish to take a step back and reconsider the potential of the platform, initially from a technological point of view. While investigating anew the capabilities of the Experience Cylinder we would like to make an exploration of the concept of immersion, using the Experience Cylinder as a vehicle for this theoretical investigation. The findings from this investigation will be used to evaluate the Experience Cylinder's technical capabilities.

In popular culture there seems to be no agreed upon definition of the concept of immersion, and it seems to be used to describe a vague ideal rather than a concrete phenomenon. In the context of popular culture it seems to be that immersion somehow relates to experiences being more realistic, convincing or more desirable as claimed in a blog about the five most immersive games:

> Immersion requires something more than simple tricks: the player has to be made to ignore all outside stimuli. The monitor/TV has to melt away, the controller has to disappear, the headphones have to turn weightless, and perception of "real" time has to be skewed. Basically, the player has to forget they're playing a game, and instead be convinced they've been put in whatever situation the game takes place. *(Little Players 2011)*

Or as in this blog site review of the ten most immersive worlds in games:

> Every seasoned gamer has an epiphany at some crucial interval during a session whether they're conscious of it or not. Usually, this phenomenon occurs when the entire game design is firing on all cylinders and your inner-self whispers, "Ok, I'm all in," and suddenly the hours just melt away. *(Hill 2012)*

In the scientific literature, various games and certain technologies are labelled as immersive. The concept is used in a number of different situations and with different meanings. Both in literature on virtual reality and literature on video games, attempts have been made to define immersion but the results seem ambiguous. Although we find the clarity of the concept questionable, it is nonetheless interesting in the sense that it entangles many aspects of our work:

the platform, the content, and how people react to it. Therefore we want to get a deeper understanding of what the concept of immersion actually means in our context. In summary, what we aim for is a simple model that places immersion alongside some of the important related concepts in our context and links them together.

We also recognize the ability of Experience Cylinder to surround the user and present an unusual screen format. The large cylindrically shaped display in combination with the infrared camera and the positional sound system makes it, in our minds, suitable for displaying virtual worlds that are interactive through physical movement. Games provide a good application area, providing possibilities for exploiting the technologies available, and being an archetypal immersive application. Games have the ability to make use of all the available technologies in a way that make them play together without the same attachment between media type and platform we see in *The Sea Stallion*. Thus in this project we focus on investigating the design and prototype implementation of an interactive game in a virtual world; in doing so we construct a model of the concept of immersion and relate our game prototype to that model.

1.1 Problem Statement

To recapitulate the previous discussion, the Experience Cylinder is a unique platform for interactive media, but its initial areas of use were limited to interactive presentation of media objects such as videos, pictures and sound. We see the potential of the cylinder to function as a platform for a wider class of interactive installations that are typically labelled as immersive, including games. However, there is no existing documentation of how to create content for the platform, let alone game-like content, and furthermore the meaning of the concept immersion is far from clear. This leads us to formulate the following problem statement in three parts:

How can the Experience Cylinder be used to support immersive game experiences?

- **What are the challenges of developing a game prototype for the Experience Cylinder, and what are the applicable solutions?**
- **How can the game prototype be evaluated with respect to the concept of immersion?**

Our approach to these questions is to develop a game prototype as a vehicle for identifying and dealing with the challenges that arise. In parallel we investigate the concept of immersion and its related terminology from which we make a conceptual model. After development we can use our model of immersion in gaming experiences to analytically evaluate our game prototypes to discuss whether the Experience Cylinder is suitable for supporting immersive games. On the basis of a discussion on the different challenges we will sum up the most important findings from our development.

From the general reader's point of view, the dissertation makes the following contributions:

- A critical discussion of the immersion concept that is intertwined with the practical part of this thesis. This is summarised in our model of immersion in gaming experiences.
- An explanation of how to develop game-like content for the Experience Cylinder.

2 Method

From the very beginning it was our intention to investigate the Experience Cylinder, with a special focus on its ability to display a 3D environment and create an immersive game experience. We have not looked for specific user needs or potential commercial applications, as this was not part of our focus.

In this perspective we would like to relate our approach to theory of evolutionary economics related to innovation. The source of innovation is said to come from either one of two driving forces: technology push or demand pull (van den Ende and Dolfsma 2005). Technology push is the situation where scientific advancement drives the development of new kinds of technology and demand pull is technological development aimed at fulfilling customer desire. Whether an innovation breaks through is a question of three factors: market opportunities that allow the innovation to spread; push, meaning the technology as fundament for the innovation; or pull, meaning the demands for the innovation.

In our situation the driving force for innovation is the technology push factor. There are no specific external demands for what we develop and therefore the project does not involve user-driven development either. In spite of this, one could argue that there is latent demand in that users are always looking for new, more exciting or novel game-playing experiences. We do not claim to lay the ground for a new technology paradigm, but simply point to the direction of the driving force for investigating the Experience Cylinder. This notion puts the technology into focus rather than specific user demands, and is reflected in our approach to the investigation.

As the cylinder has not been used for the purpose of displaying a 3D environment before, we found it necessary to first prove that the concept was feasible. We searched for literature on installations that had similarities to the Experience Cylinder, and through this research found software that allowed us to test the concept.

Based on the results of this proof of concept and the literature we had found, our work divided into two parts. Our work in creating a game prototype was one part. This involved solving emergent issues relating to utilizing hardware and software in the Experience Cylinder as well as creating a software prototype. The other part was the methodological issue of how to evaluate the prototype. Evaluating immersion through user tests was discussed, but ultimately deemed impractical. Evaluation based on user tests would require a testable product far ahead of the project deadline. We decided on the more feasible approach of creating a conceptual model, based on immersion literature, through which we could evaluate our prototype.

3 Conditions

This project has taken place during the spring 2012 in the basement of building 40 at RUC. The Experience Cylinder is an ongoing project with several actors both researchers and technical staff. During the development period we have held several presentations for externals and meetings with the project steering group about the development of the Experience Cylinder as an integrated research and education platform. The majority of the time the Experience Cylinder has been a stable construct, but the discussions with researchers and staffs has been focused on the changes to the different technologies.

4 Proof of Concept

To determine, if it was possible or relevant to construct a 3D game prototype in the Experience Cylinder, we first had to examine what a 3D game would look like in the Cylinder. In order to do this we had to find existing software, which would allow us to show some kind of 3D environment across multiple screens and could be configured to our specific needs. We decided to start on the basis of existing 3D games, and to investigate whether anyone had modified such a 3D game to allow for multiple screens.

We found a paper by Jeffrey Jacobsen, who had configured the 3D game Unreal Tournament to work in a CAVE (Jacobsen 2005a). The modification is called CaveUT, based on the abbreviation of the game they used. Their modification was freely available on the internet (Jacobson 2005b). Using this software and a number of Unreal Tournament game copies on an equivalent number of networked computers, we created two test setups. We started with a small setup consisting of four computers with normal monitors; with the purpose of testing to what extent the software could be configured to our setup. The software proved to be highly flexible, and could be configured to more than the four displays typically used for the sides of a CAVE. Additionally the positions of the monitors were configurable as well to allow a setup based on - for example - six displays in a circle rather than four projectors in a square. This showed us that it was possible to utilise the software for the specific projector setup in the Experience Cylinder. However it also showed us, that the game design was limited to primarily using one primary screen, with the additional screens only providing peripheral vision.

For the next test we acquired enough computers to test the software in the Experience Cylinder, and consequently configured the software for that specific setup. The test was a success, as it showed us that it was indeed possible to show a 3D environment in the Experience Cylinder. Subsequently, we contacted the creator of the CaveUT modification to inquire about any further development on the modification. He informed us that the team behind CaveUT had switched to another software solution, a game development environment called Unity3D.

To sum up our proof of concept work, it had shown us the feasibility of displaying 3D games in the Experience Cylinder, and our contact with the CaveUT author had given us a lead on a potential game development platform.

5 Investigation of Immersion

This chapter deals with the investigation of the concept of immersion. In the introduction we noted that the concept is used with different meanings and in different contexts but there also seems to be some kind of common denominator. Games and virtual reality make use of the concept but since we have a combination - a game in a virtual reality setting - we need to investigate immersion in both areas on a conceptual level. We do this to determine to what extent it can be utilised in an evaluation of experiences in the Experience Cylinder.

In the investigation of the concept immersion a number of related concepts emerged. In order to follow the unravelling of immersion in the context of games and the context of virtual reality we will first present these concepts separately. After this we give an introduction and definition to what a video game is, based on the definition of the even broader concept *game*.

With these sections as a foundation we can begin to introduce the meaning of the concept immersion in context of games. We do that by comparing four articles that specifically try to do the same - namely to make a model of immersion in games. We extract the parts we find that the models have in common. After this review we introduce the concept of virtual reality, before we explain what the concept of immersion signifies in the context of virtual reality.

Finally we close the section by summarising the most important differences between the two areas.

5.1 Central Concepts Related to Immersion

The central concepts that we refer to are *Flow*, *Presence* and *Cognitive Absorption*. We describe the concept of flow since much of the literature on immersion in games refers to this concept. Presence is a concept we find in the literature on virtual reality; however it is more specific than the descriptions of immersion in general. One could argue that this concept requires as much attention and work to get around as immersion does, but the difference between the concepts is that immersion is much more widely but differently used in both the game literature as well as in virtual reality literature. The concept of presence is primarily used within literature on virtual reality. As in literature on the concept of immersion in games, presence is divided into different subtypes. These subtypes of presence seem more consistently described as opposed to the subtypes within the literature on the concept of immersion. Cognitive Absorption is a concept that is developed in a different application context than games but describes some of the same effects that we find in descriptions of immersion.

5.1.1 Flow

This concept was introduced by the Hungarian psychologist Mihály Csíkszentmihályi and defined as “*the mental state of operation in which a person in an activity is fully immersed in a feeling of energized focus, full involvement, and success in the process of the activity”* (Csíkszentmihályi and Csíkszentmihályi 1998).

The experience of flow is characterised by a continuous match between a person’s level of skill and the challenge at hand. That the experience is continuous means that as the task at hand gets harder, the level of skill supposedly increases, and therefore the level of challenge should rise accordingly in order to maintain the state of flow. If this match is not made, the person will either experience boredom in the case where the challenge is too easy because of high skills or anxiety because the level of challenge exceeds the level of skills.

The effect of the state of flow can be recognized by the sense of feeling in control, that it removes the awareness of everyday life from consciousness, and the sense of duration of time is altered. Flow is often depicted as in this illustration:

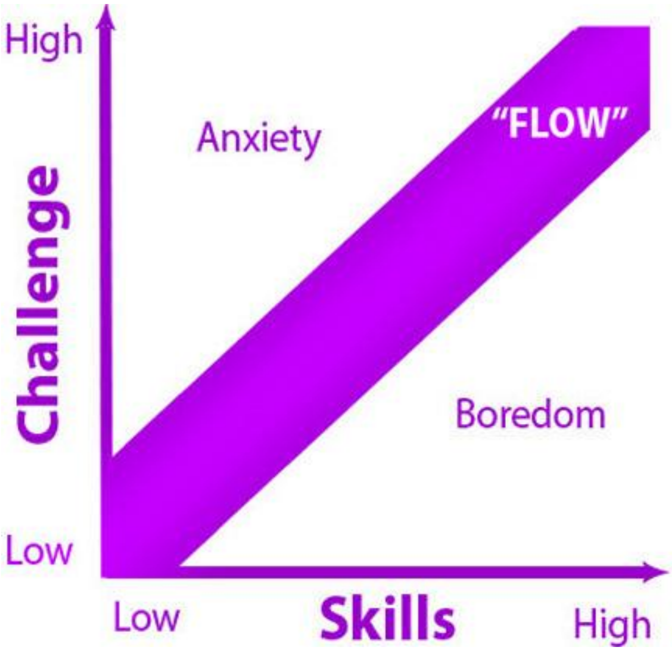


Illustration 5-1: Flow can happen in the situation of learning some new skill - say building LEGO. First you have to figure out how bricks are attached together and then you can begin to construct more and more advanced combinations of bricks. If the learning curve is right the incremental process of learning that skill is what flow is all about. (Hills and Hills n.d.)

5.1.2 Presence

As we approach the literature on virtual reality and the attempts to describe the effects of the technologies we again encounter the concept *immersion.* However another concept that seems to be related is *presence.* Before we begin to examine the concept of immersion in virtual reality literature, we here give a short introduction to the concept of *presence*.

Presence is described popularly as the feeling or *sense of being there* (Sheridan n.d.) and *the sense of feeling present*. Academically presence is characterised as *"the extent to which media represent the world"*, (Heim 1993). When the media are convincing and you cannot distinguish representation from the real world it is total presence. We simply give a brief overview of some of the different definitions to the concept of presence and point towards a definition that we can use in the making of a model.

Originally the term presence seems to be derived from the research on *remote operations*. Presence in this context concerns how to control a machine remotely and out of sight, but experiencing a sense of being at the same place as the machine in spite of the distance (Sheridan n.d.). It is postulated that this sense is made by a mental representation in the mind of the operator. The preconditions to experience such a sensation are that there is no *lag* between input and feedback, and that the remote *manipulator* - the terminal - displays the remote site in a way that matches the real sites. To explain the broader lines in presence we find the division in *personal, social,* and *environmental* presence as a good way of presenting the different related aspects to the overall term presence.

Personal Presence............concerns the user's psychological mental state explained as *the suspension of disbelief* or the *sense of being there* (Sheridan n.d.). If a person experiences presence it signifies that the virtual environment is perceived as real as the counterpart environment it represents. This understanding of presence is slightly different from the concept it is derived from - telepresence - in the sense that presence in virtual reality context can be about environments that do not have a real world counterpart.

Social Presence................emphasizes social interaction and communication within a virtual environment setting. If users meet other people in a virtual environment and can communicate with them it "*will help add meaning to the world and further increase the sense that the virtual environment is more than simple images and thus make it more real*" (Nunez 2003). In other words social communication and interaction is familiar to most people. If you can communicate with other people through the virtual environment in a way you are used to, it will be easier to accept that the medium is not real but computer generated.

Environmental Presence .. concerns perception of virtual objects in the virtual environment where perception is about relating information about the objects represented and how they can be interacted with. If we imagine interaction as a way of *communicating with things*, the understanding of environmental presence is similar to that of social presence. If a person finds the way of interaction with virtual objects intuitive, in the sense that they are similar to how they can be interacted with in the real world, it is easier for the person to accept that the medium is not real. Experiencing environmental presence signifies the recognition of virtual objects and their nature as similar to real world objects.

5.1.3 Cognitive Absorption

A third relevant concept to introduce is the concept of Cognitive Absorption. It belongs to research on use and adoption of information systems and is grounded in the research on technology acceptance. Cognitive Absorption is originally defined as *"a state of deep involvement with software*" (Agarwal and Karahanna 2000). The research is primarily focused on utilitarian information systems, but it is also relevant to consider Cognitive Absorption in a hedonic (pleasure-oriented) context. The field of hedonic information systems is covering, among other application forms, video and computer games in the sense of home and leisure activities (Weniger and Loebbecke 2010).

Cognitive Absorption is based on five concepts: *temporal dissociation* which is the sensation of losing track of time, *focused immersion* as the *"total engagement where other demands are ignored"*, *heightened enjoyment* accounting for the pleasurable part of an interaction, *control* as the sensation of being in charge, and lastly *curiosity* as the degree of *"an individual's sensory and cognitive curiosity"* (Agarwal and Karahanna 2000). The interesting aspect of the conceptualization of cognitive absorption is that it describes the user's sensation about an experience.

Absorption connotes that a person unwillingly can be engaged in an activity. We find the word *involvement* more appropriate in the sense that is signifies that a person must be *internally motivated* before s/he can be absorbed. This is opposite to presence where motivation is not a factor.

5.2 Game Immersion

Having explained the concepts flow, presence, and cognitive absorption, we want to apply the concept in a certain situation. We introduce and explain relevant parts of the domains of games, before we investigate the concept of immersion in this context.

We begin by introducing the concept of games as well as video games because this is the application type area we investigate. We can use concepts describing central aspects of games to make connections between central game features and their effects on the person that plays them. These connections become clearer as we treat the just mentioned concepts in combination with the following investigation of immersion.

5.2.1 What is a Game?

In the paper *The Game, the Player, the World – Looking for The Heart of Gameness,* Jesper Juul attempts to come up with a definition of what a game is (Juul 2003). His initial assumption is that a good game definition should contain three elements:

The *game* .... as *“a kind of systems set up by the rules of the game.”*
The *player* .... as *“the relation between the game and the player of the game.”*
The *world* .... as *“the relation between the playing of the game and the rest of the world.”*

These elements are connected to present a single definition for a game as:

> A rule-based formal system with variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable.

Juul’s definition describes six features a game must exhibit in order to be a game and anything that has these features is a game. They are:

- Fixed rules
- Variable and quantifiable outcome
- Valorization of outcome
- Player effort
- Attachment of the player to the outcome
- Negotiable consequences

In the following we explain each of these features with the classic board game Ludo as an example:

Fixed Rules .... Fixed rules are fundamental to any game and they have to be unambiguous. Rules are what govern the change in the game and player actions change it from state to state according to these rules. In Ludo a player has to make decisions about which tokens to move based on the dice roll. The rules dictate the possible moves.

Variable and Quantifiable Outcome .... Variable and quantifiable outcome means that as an effect of the rules the game must be able to provide different outcomes in terms of winning and losing conditions. In Ludo the variable outcome is determined by the players finishing place. The winner is the one who gets all his tokens to the finishing square. As the game is turn-based there cannot be two players to achieve this condition simultaneously and the outcome is therefore unambiguous and quantifiable.

Valorization of Outcome .... In addition games must afford valorisation of outcome which means that in Ludo players should automatically strive for certain outcomes - the winning conditions!

Effort .... Players should also put *effort* into achieving this condition through invested time and energy.

Attachment to the Outcome .... As a natural cause of effort, players should feel *attachment to the outcome* meaning that if a player wins a game of Ludo it must somehow be a result of this effort as well as skills.

Negotiable Consequences .... The last game feature is *negotiable consequences* which is the option for any game to be applied real-life consequences i.e. betting money on the outcome of a game. In any event the question of honour is at stake. A final remark to this game model is the question about borderline cases such as *rules governing the stock market* or democratic elections. They could also be considered games within this model, but is not included because the real-life consequences are non-negotiable.

5.2.2 What is a Video Game?

We could have made a straightforward extension of the definition of games to video games but we would like to present another definition specifically of video games. Nicolas Esposito, researcher in video games at University of Technology Compiégne suggests:

> A videogame is a game which we play thanks to an audiovisual apparatus and which can be based on a story. *(Esposito 2005)*

It adds to the definition of what a video game is but we suggest a slightly modified definition where *apparatus* is exchanged with *computing device*: a video game is a game which we play thanks to an audiovisual computing device and which can be based on a story.

The addition of video to game is simple to define as something that falls into the above description and is implemented on a computer. The definition of video games adds extra dimensions compared to the understanding of games which we now explain. We consider the visual part of audiovisual trivial as all video game devices contain a display of some sort.

Sound in video games can roughly be divided into two kinds: one is effects that are coupled to actions in the game and the other is theatrical sounds or music. Sound effects are used to direct attention and accentuate the meaning of a given action and music is very useful for creating atmosphere.

Audiovisual computing device can be divided into four different groups of devices: desktop computers, game consoles, handheld consoles including smart phones, and LBE-machines (Location-Based Entertainment). This division is blurred by the fact that desktop computers slowly are replaced by laptops and tablets that can be considered handheld, and the latest generations of game consoles can be considered multi-media devices since they support internet browsing and media center capabilities.

The video game definition leaves the narrative part optional. Within our previous mentioned board game example Ludo it makes sense, but video games without at least an introductory story is rare. A story is an important factor of any modern video game but in the game genres simulations and sports it is often left out. To finish this explanation of the definition of a video game, we would like to quickly introduce some of the main genres of video games (Wikipedia 2012).

Action Games.................. are characterized by their focus on tempo and skills such as: quick reflexes, accuracy, and timing. These skills are required by the player in order to overcome obstacles.

Adventure Games............ focus mostly on puzzle-solving, appealing to a player’s problem solving and logical skills. Adventure games often focus on the specific game world and the best known examples are the Myst game series.

Role-Playing Games.......... build on the heritage of the pen-and-paper based Dungeons & Dragons games. Most famous are the Diablo games made by the company Blizzard. The player chooses a character with special abilities that leads to different ways of playing the same game. The objective is to use these abilities wisely and improve on them during the game in order to achieve the main goal.

Strategy Games............... are a genre in which a player, on the basis of collecting a resource, builds an army consisting of a number of units with certain capabilities. The challenge is to balance resource management and composing an army of suitable types of units and attack the opponent(s) before the opposite happens.

Simulation Games .... can be considered a borderline case, since the simulation can be so realistic that the application is no longer considered a game. A game typically has a focus on the interesting parts of the real world and leave out the boring parts. The simulator on the other hand focuses on realism to the extent that repetitive or trivial parts from real life are not avoided. As an example a flight game could possibly consist of immediate action of dog-fighting. Oppositely the flight simulator could consist of: a full training program before any mission, realistic waiting time, briefing, and debriefing. Additionally, rules and outcomes are not always defined as clearly as with the other genres. In a flight simulator for domestic use on a computer, the purpose might be obvious in terms of successful take-offs and landings, but it is up to the player to define the main goal of the entire game, as simulation games are open-ended.

Having described what a video game is, and given examples of different genres, we continue with the investigation of the concept of immersion in the context of games.

5.2.3 Four Subtypes of Immersion

This section deals with the subdivision of the term immersion as we have seen it described in literature about games. We have chosen four different articles that share the same goal - to divide the concept of immersion into subtypes, resulting in four different models for immersion. It is these models that we compare. The articles are:

- *Immersion, Engagement, and Presence* (McMahan 2003).
  Alison McMahan has a PhD in film studies and is co-author to the book *In the video game* in which the article appears.

- *Fundamental Components of the Gameplay Experience: analysing Immersion* (Ermi and Mäyrä 2005).
  Frans Mäyrä is a professor of information studies interactive media at the University of Tampere.

- *Immersion Revisited: on the Value of a Contested Concept* (Thon 2008).
  Jan-Noël Thon has a PhD in transmedial narratology and is a research associate at the University of Hamburg, Department of Media Studies.

- *Postmodernism and the Three Types of Immersion* (Adams 2004).
  Ernest Adams is a game design consultant. His article appears in the game magazine *Gamasutra*.

We have found that immersion in game context can be divided into four subtypes, namely one concerning the *spatiality* of the game world, the *gameplay*, the *narrative,* and *social* aspects. The following section is structured with a subsection of each of the four types. Each subsection treats each of the articles that mention the related type of immersion.

○ Spatial immersion

We find similar recognitions of what we call spatial immersion in three of the studied articles. First McMahan points it out as realism and perceptual immersion. Realism is defined as: *"how accurately does the virtual environment represent objects, events and people"*. In this sense realism is a combination of social and perceptual realism. It depends on whether the game depicts the real world both regarding interaction forms that imitate the real world as well as photo-realistic depiction where the game objects look realistic. The perceptual realism is very close to what we within game immersion label spatial immersion. Perceptual immersion is *"blocking as many of the senses as possible"* but should rather be understood as *saturation* of the senses since the virtual environment is stimulating the sensory apparatus in order to shut the external, non-mediated world out.

In (Ermi and Mäyrä 2005), this dimension of immersion is referred to as the *audiovisual quality and style,* exemplified as *"good looking graphics, well-functioning camera angles".* However aesthetics are perceived individually from player to player. The dimension is named: *sensory* immersion and is described as *"…impressive, three-dimensional and stereophonic worlds…"*

In (Thon 2008)*, spatial immersion* is described with the focus on *the virtual game world,* i.e. the game space in which the player moves around his/her virtual character, called an avatar. Spatial immersion is concerned with the game space that is manoeuvrable by the avatar. That leaves out both the real world of course and the part of the game world that cannot be reached by the avatar. Further spatial immersion considers the part of the game space that is relevant for interaction. With our own words spatial immersion signifies both a player's attention towards the virtual game world and how convincing it seems.

Recapitulating on the different descriptions of spatial immersion we can say that they are somehow connected to the game space, the visual effects and the sense of realism. The descriptions are not totally overlapping but treat the same aspects of games. Spatial immersion could be interpreted as the immediate ability to impress the player with a credible game world and saturate the player's sensory apparatus in order to shut out real world stimuli.

A games audiovisual quality and style are the structural parts of a game that are related to the effect a player is experiencing referred to as spatial immersion. We find that the concept of environmental presence is related to spatial immersion.

○ Immersion in gameplay

If one considers a game that is stripped of fancy graphics, a narrative to link different game events together and any social relation to other players, all that is left are the challenges of the game. It could be the challenge to gain as many points to beat a high score before the game is over. We call this concept *gameplay* which signifies what a given game is essentially about. McMahan is quite brief about this important foundation of games. We recognize it in the definition of *psychological immersion* where it is shortly described as *"the user's mental absorption"*, but that fits the overall vague term immersion related to games in general.

In (Ermi and Mäyrä 2005), it is described as the *level of challenge* and has to do with balance and the way of advancing and succeeding in the game. However this falls further into two categories: *sensor-motor abilities* that require fast actions and *cognitive challenges* that require thinking. The dimension is named *challenge-based immersion* and is defined as the *"...satisfying balance of challenges and abilities"* and accounts for low-level sensory-motoric skills as well as high-level strategic and logic problem solving skills.

Also in (Thon 2008), we find a similar concept namely *ludic* immersion. It is the focus on the interactive part and is about the possibilities of interaction. It is about *"…the player's actions that result in actions of the avatar and/or a change of state of the various objects…".*

Only Adams delves deeper into the two kinds of game-play that Mäyrä distinguishes but treats similarly. He calls it *tactical* and *strategic* immersion. *Tactical* immersion is popularly described by statements as *being in the zone*, *being in the groove* and as a *meditation-like state*. The focus is on short path challenges that need immediate and quick action from the player but are also easily solved individually. This stands opposite to *strategic* immersion, which is described as an activity of *observing, calculating, deducing* and exemplified by playing chess which requires problem solving on an abstract level. It is also stated that the challenges must be enjoyable without depending on chance. This in combination with the *tactical* immersion is very close to what we have previously described as *flow*.

All four articles elaborate on this central aspect of games without labelling it gameplay. This seems fair in the sense that gameplay in our understanding is a structural part of a game itself, and the mental state a player is experiencing is similar to the state of flow. We think of gameplay as *an objective structure of a game*. It is the foundation for a player's involvement and chance of entering the mental state of flow.

○ Immersion in Narrative

Apart from McMahan all of the mentioned articles recognise the narrative as something a player can get immersed in. In (Ermi and Mäyrä 2005), it is all aspects of player imagination related to the game. In the description it could seem to overlap with some of the descriptions of spatial immersion, as well as some of the social aspects that the other models include. Imaginative immersion is about the virtual world of the game, the characters and storyline, and somehow the ability to do things not possible - or acceptable - in the real world. It is described as the way in *"...which one becomes absorbed with the stories and the world, or begins to feel for or identify with a game character".*

The narrative dimension is described in (Thon 2008) as the unfolding of the story in a game. A distinction between narrative events and ludic events is made to underline the difference between the kinds of immersion related to such events: *"Narrative events are determined before the game is played and are presented using the various techniques [cut-scenes and predetermined sequences]".*

This is opposed to ludic events that are determined as they are played out. Narrative immersion in this case is considered to be about the whole game world – meaning also the places the avatar cannot reach. Further the description of narrative immersion contains two subtypes of immersion referring to temporal and emotional immersion. Temporal immersion is described as the desire to know what happens with the story plot whereas emotional immersion is concerned with the experience of empathy related to the character's fate.

A similar categorisation is found in (Adams 2004), where narrative immersion is put plainly as the player's *"care about the characters"* and the urge *"to know how the story is going to* *end".* Despite similar word usage the concern for immersion in narrative spans from the player's attention to the story in a game to include emotional attachment to characters in the game story. We find that the concept of *involvement* is related to narrative immersion understood as descriptions of emotional connections to in-game, non-human characters as well as the game-story.

○ Social Immersion

The last dimension may also be the broadest one and the one that is hardest to embrace in a coherent description because the different sources define it so differently. We begin with three different concepts from (McMahan 2003), which all relate to what we label social immersion. First we have social interaction described as the player's sense of *togetherness* or being with someone even though it is only in a game. This is achieved if *"…alterations of the environment caused by the actions of one participant are clearly perceived by the other participants…".*

Notice how close the formulation is to the meaning of social presence. We will not go deeper into the formulation but remain satisfied with the focus on social interaction with other players. Further the social dimension includes the relation to a player's avatar as well as non-playable characters. This overlaps with empathy in narrative immersion. Thirdly immersion in an *intelligent environment* can presumably cause a sense of presence: *"a sense of presence can result from users responding to the computer itself as an intelligent, social agent".*

The social dimension is also found in (Ermi and Mäyrä 2005) as the social context in which the game is played, and in (Thon 2008) the dimension is described against the backdrop of narrative immersion. With this type of immersion we are talking about inter-contextual immersion understood as *"communication and social interaction of the players with each other"*. This is closely related to narrative immersion as this mentioned communication *"may additionally intensify players' experience of narrative immersion".*

Immersion in social context is fairly clear as long as it only concerns relationship between human players experienced through the game. As mentioned the concept is blurred as social context also accounts for a player's relationship to in-game characters as this overlaps with immersion into narrative. We find that immersion in social context is related to the concept of social presence.

5.2.4 Overview Table

In the Illustration 5-2 below we have placed the four types of immersion. The horizontal axis is divided into two parts: aspects of immersion within the game and aspects of immersion external to a game. It is not to be taken too literally but is simply a way of separating the concepts. On the vertical axis we have placed the four different articles and their respective way of naming the different kinds of immersion. The top-most line is our subdivision and under that is (Ermi and Mäyrä 2005), and so forth. The overview is to give an indication of how the different divisions fall within the same categories without us having done anything to quantify each of the immersion types.

Each of our four subtypes of immersion have been identified in at least three out of four models. Although the individual subtypes are not called the same across the articles we have presented arguments for their similarity. We find that immersion can be divided into four different types within game context.

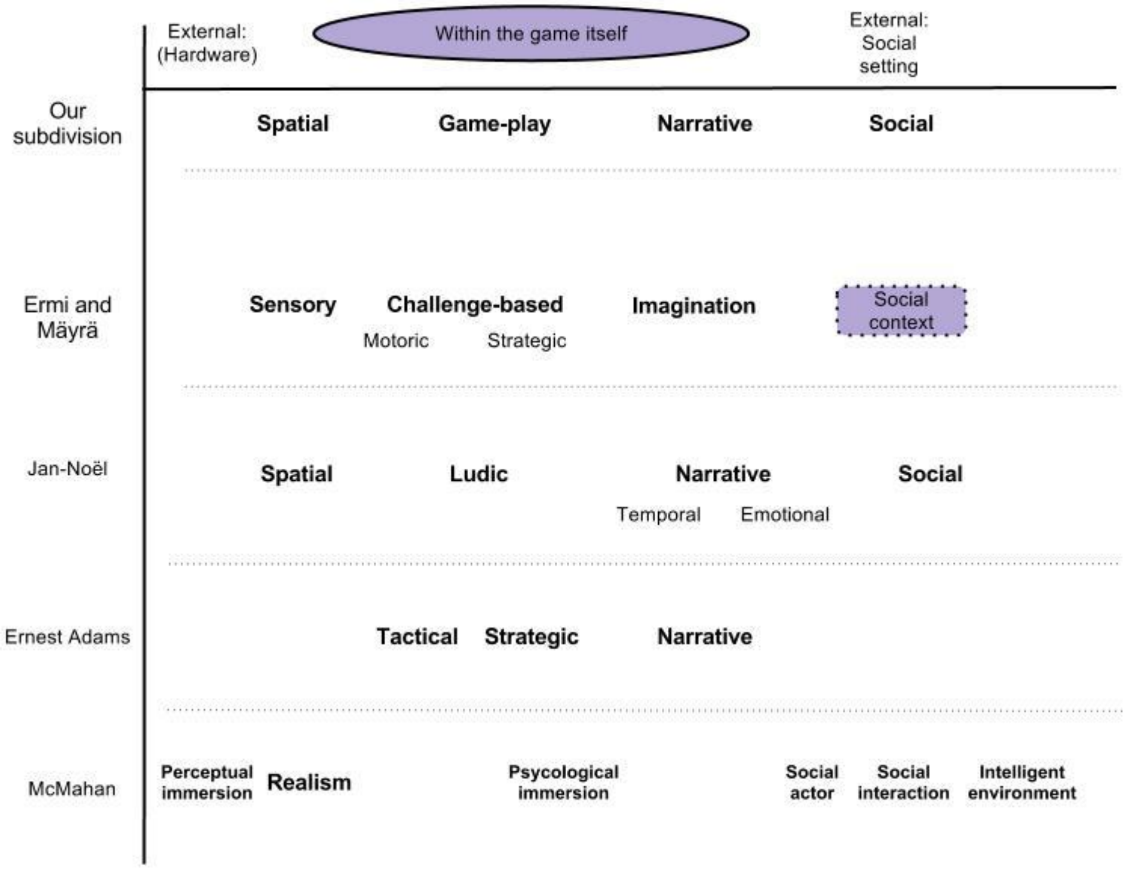


Illustration 5-2: The horizontal axis is divided into two parts: aspects within the game and aspects external to a game. It is not to be taken too literally but is simply a way of separating the concepts. On the vertical axis we have placed the four different articles and their respective way of naming the different kinds of immersion.

5.3 Summary

We summarize from our investigation that *spatial immersion* is related to the concept of *environmental presence;* that *immersion into gameplay* is related to a player's *involvement* and chance of entering the mental state of *flow;* that *immersion into narrative*, understood as descriptions of emotional connections to in-game characters as well as game-story, is also related to a player's *involvement*; and that *social immersion*, understood as relations to other human players, is related to the concept of *social presence*.

This also means that each type of immersion is related to a corresponding inherent game structure that is the foundation to cause immersion.

5.4 Virtual Reality Immersion

The Experience Cylinder can be considered to belong to a class of system known as a CAVE which is a virtual reality technology. In the following section we provide a brief introduction to the development of central virtual reality technologies to show how the connection between the Experience Cylinder and virtual reality emerges. The section is followed by a review of the term immersion in virtual reality context.

5.4.1 What is Virtual Reality?

Coined in 1987 by Jaron Lanier, virtual reality is a quite broad concept that is originally coined as a self-contradiction on purpose. How can anything be both virtual and real at the same time? a rather abstract definition of the concept describes virtual reality as "*an event or entity that is real in effect but not in fact*" (Heim 1993). Often though, virtual reality is defined exclusively in the combination with a specific technology.

Morton Heilig conceptualized the *Sensorama in 1952* - a contraption that consists of: a stereoscopic 3D display, fans, odour emitters, stereo speakers, and a moving chair (Heilig 1962). In the Sensorama the participant could experience a bicycle ride through Brooklyn. The invention is in retrospective considered the first virtual reality technology.

Illustration 5-3: Morton Heilig conceptualized the *Sensorama in 1952* - a contraption that consists of: a stereoscopic 3D display, fans, odour emitters, stereo speakers, and a moving chair. In the Sensorama the participant could experience a bicycle ride through Brooklyn. The invention is in retrospective considered the first virtual reality technology.

The inventor of the head-mounted display, Ivan Sutherland, bases virtual reality on three features: being indiscernible from the real world, based on a computer that generates the world in real time, and the ability for the user to interact and manipulate with virtual objects in an intuitive way. The head-mounted display which was invented in 1968, see Illustration 5-4 below, shuts out the normal view and displays a computer generated one for the viewer.

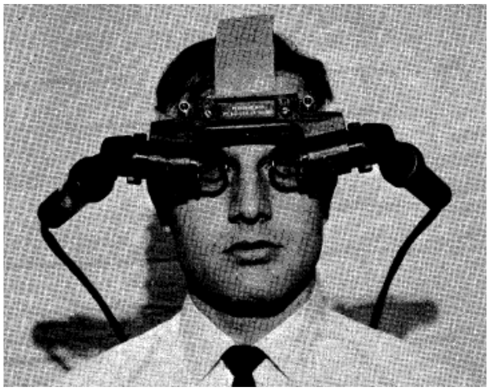


Illustration 5-4: Head Mounted Display invented by Ivan Sutherland, provides small monitors to be positioned in front of the eyes.

This technology was later combined with a digital glove or hand-held device used to manipulate virtual objects and provide sensory feedback to imitate the feeling of touching these virtual objects. This accounts for the third part of the virtual reality definition.

In 1985 Myron Krueger built an interactive environment called VIDEOPLACE (Krueger, Gionfriddo and Hinrichsen 1985). A digital representation of the participant's silhouette is projected on the screen. Through this the participant experiences the ability to interact with virtual objects.

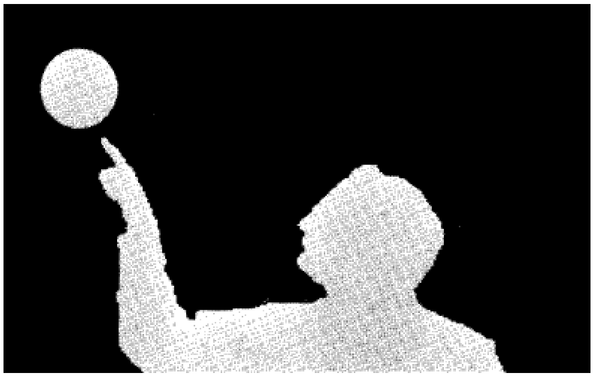


Illustration 5-5: In VIDEOPLACE by Myron Krueger a person's shadow is computer generated and projected onto a screen. Virtual objects are manipulable through the virtual representation of the person. "The circle can be pushed".

This could be considered the first instance of what is later called a CAVE. CAVE is originally a recursive acronym for *cave automatic virtual environment* where projectors display a computer generated 3D environment on the inside of a box-shaped room. The point is to surround the viewer with visual input and thereby also block out external visual stimuli. Stereoscopic 3D is utilised in order to enhance the sense of depth in the picture and motion tracking of head-movement is used to adjust perspective according to the viewer's point of view. The goal is the same as with the head-mounted display and gloves but the equipment is less obtrusive.

5.4.2 Immersion and Presence in Virtual Reality

In the context of virtual reality, we once more encounter the use of the word immersion. In (Bowman and McMahan 2007) it is stated that: "*Immersion refers to the objective level of sensory fidelity a virtual reality system provides.*". Furthermore they elaborate that immersion "*...is objective and measurable—one system can have a higher level of immersion than another.*".

The level of immersion is described as *"...how close the system's visual output is to real-world visual stimuli."* and a factor of many components such as for example the field of view, size, and resolution of a display. (Bowman and McMahan 2007).

A checklist of what we could call immersive components defines the level of immersion for the technology in question. Since we do not attempt to measure immersion in our context we will not delve deeper into the understanding of the objective and measurable attributes of immersion. Just notice that we did not find any such checklist that ranks technologies on a scale.

When it comes to defining presence within virtual reality context it is clear that the concept has been developed within virtual reality research. The definition and meaning of the concept seem clear:

Presence is *"…an individual and context-dependent user response, related to the experience of "being there"."* (Bowman and McMahan 2007). And the goal is to, in the user's mind *"let the user experience a computer-generated world as if it were real - producing a sense of presence, or "being there"* (Bowman and McMahan 2007). Hence the concept of presence can be understood as: "*a user's subjective psychological response to a VR system."* (Bowman and McMahan 2007). Different users can experience different levels of presence with the same virtual reality system, and a single user might experience different levels of presence with the same system at different times, depending on state of mind, recent history, and other factors. This is very identifiable as the kind of presence we have previously described.

6 Building a Model of Immersion

The conflicting understandings of the concept of immersion in game literature and literature on virtual reality are problematic to combine in a coherent manner. In order to combine the two areas we need one that takes into account both games and technology used to play the game.

In this section we describe the motivation for creating a model of immersion that attempts to unify immersion theory. The gradual construction process is described to show the change in our understanding of immersion as we attempted to construct our model.

To bridge the gap between the platform - the Experience Cylinder - and the content type of games, we propose this model as a means to get an overview of immersion in this context, as well as a proposed tool with which to evaluate experiences in the Experience Cylinder. We by no means believe this to be an exhaustive model covering the present range of topics.

6.1 Construction

The starting point of our modelling efforts was game immersion literature. We started by identifying the different definitions of the concept of immersion, and more specifically how the concept was subdivided in the individual texts. The purpose was to find common ground between the different articles. Everyone seemed to know just what immersion was about, but there seemed to be no generally agreed upon definition. This became especially noticeable when attempts were made to divide immersion into subcategories.

We decided to approach this systematically, by creating a table of the different articles that identified different types of game immersion, Illustration 5-2 on page 26. After this we tried to meaningfully categorize how the different types of immersion were subdivided and how they were ordered among themselves. We tried placing them on a scale ranging from *instinctive* at one end of the scale to *cognitive* at the other end. The focus of this division was mainly on the experience of the user. *Instinctive*, relating to the sense of a type of immersion that was mostly related to immediate responses and sensations (action), and *cognitive* in the sense of a focus on deliberation and higher cognitive functions (puzzle, problem-solving, complex narrative).

To some extent, this categorization was possible see

Illustration 6-1 below, but ultimately proved too simplistic by trying to list immersion types in a specific order. Some immersion types clearly belonged on a certain part of the scale, while others seemed to belong in several places. Additionally, not all articles that dealt with immersion incorporated a social aspect in their considerations and the social aspect was difficult for us to place in this form. So the original idea of division was modified slightly, to include a range for each type of immersion, see Illustration 6-2 below.

This model seemed to better explain the multi-faceted aspects of some types of immersion, but we had no scientifically sound way to quantify and decide the length or placement of the different bars, other than a vague feeling of how we felt the bars should overlap. Additionally, we still did not have a way to properly place any social aspects of immersion, as the instinctive / cognitive scale seemed more focused on an individual's experience without regard for other people.

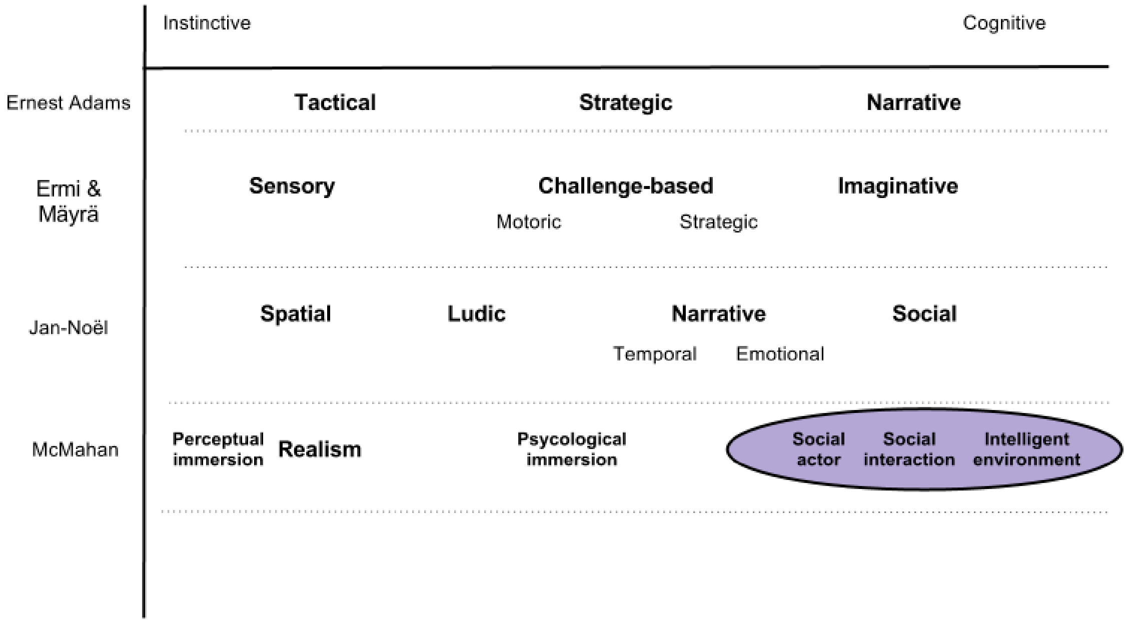


Illustration 6-1: This illustration shows a division of immersion into instinctive and cognitive. Instinctive, relating to a sense of immersion mostly related to immediate responses and sensations, and cognitive in the sense of a focus on deliberation and higher cognitive functions.

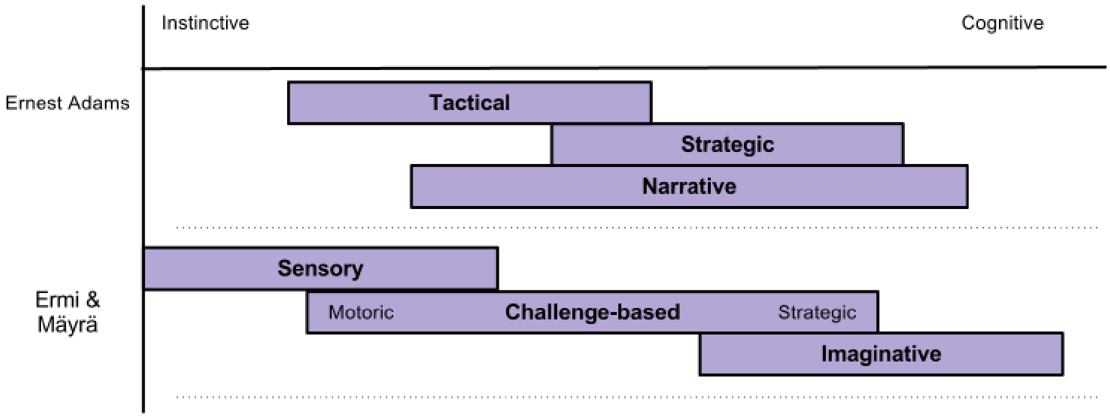


Illustration 6-2: An example of another way of placing the different types of immersion on the instinctive cognitive scale in an attempt to visualize that different types of immersion can span across a range of the instinctive-cognitive scale.

Being dissatisfied with our models we began looking for more fully explained definitions of immersion, presence, flow, cognitive absorption, and involvement. The hope was that a more thorough explanation of the terms would aid us in creating a unified model. We came across a five-page publication by Mel Slater, a professor specialised in virtual environments, in which he discusses the general confusion of the terms he experienced at a conference he had just attended (Slater 2004). He subsequently gives his definitions and understanding of the different terms, and how they relate to each other, to the experience and to the user. Mel Slater's distinction between the form of an experience and the content in the experience helped us understand some of the disparity between immersion theories from the game design field and presence theories from the virtual reality field.

From our cursory overview of the literature, immersion theory from the game design field focuses largely on the content of a given experience and rarely includes considerations about the platforms effect on overall game experience, while presence theory from the virtual reality field largely focuses on the form of the experience and avoids making generalizations about the content.

On the basis of the form and content division, we created a new model which we gradually refined.

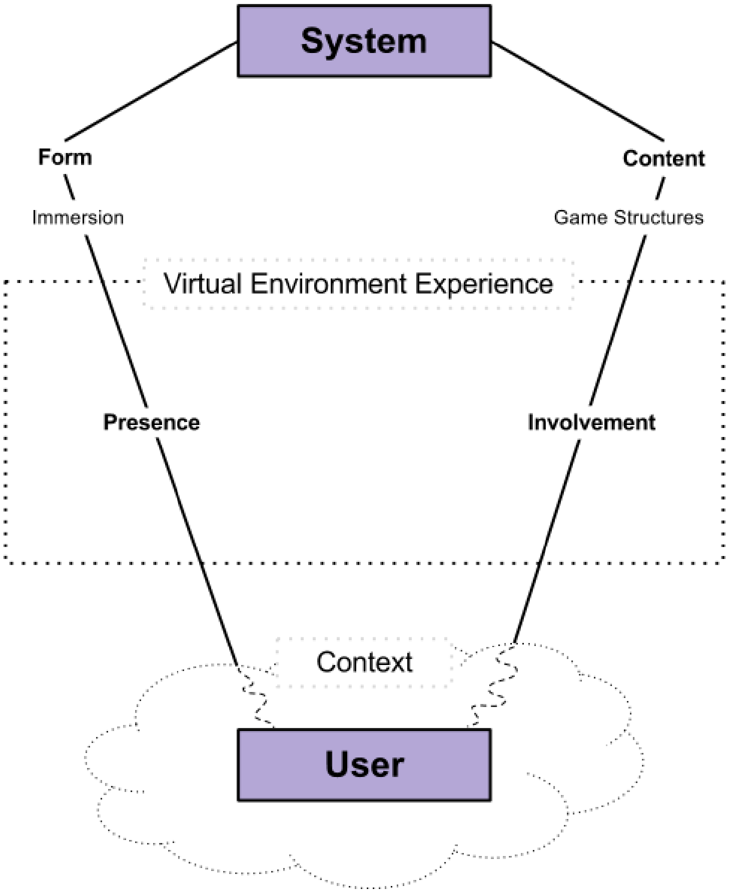


Illustration 6-3: This illustration shows our proposed model of a virtual environment experience. It links user to form and content through a certain context and points out the two different ways one can be engaged in the experience as a combination of the sense of presence and involvement. Immersion and game structures refers to concrete properties of form and content respectively.

6.2 Explanation of the Model

In order to concatenate the two different branches on immersion, we built the virtual environment experience model, see Illustration 6-3 above. It links user to form and content through a certain context and points out the two different ways one can be engaged in the experience as a combination of the sense of presence and involvement. Involvement being considered similar to the concept of immersion within game literature and presence as defined in virtual reality literature. Immersion and game structures refers to concrete properties of form and content respectively.

At the top of the model, we have the system. This is a combination of all hardware, software and other artefacts that are used to convey a game experience in a virtual reality setting to a user. The parts of the system that are visible to the user can be roughly divided into two categories: form and content. The form has to do with the direct appearance of the system to the user. This covers screen size and shape, sound system, and controls. But that is not simply to say that form has to do with hardware. The graphics engine used also concerns form, as this decides how things are presented on the screen. But the graphics engine makes little sense by itself, as does the rest of the form, without suitable content. Content has to do with what is presented to the user. It is hard to define what exactly makes up the content of an experience, so it is easier to explain through examples.

Let us take a very simple experience: reading a book. The physical book itself, and partly also the letters themselves are part of them form. The story which is presented through the book is the content. Books also fall into genres. Different genres have different ways of plot construction and different means to attract the reader's attention. Games also fall into genres and have the same ability to attract attention. We refer to these aspects of games as *inherent structural elements.* These elements are what are common to different games within the same genre both in terms of game world, gameplay and narrative. If we again consider the book analogy, the inherent structural elements of narrative in a classic fairy-tale is the invariable plot development. The main character must defeat evil in order to achieve greater good. To sustain the reader's attention the fairytale uses descriptions of the supernatural.

By the classic understanding of immersion we have adopted that a book *cannot* be considered immersive. This is because a book does not have the ability to saturate the sensory apparatus and therefore neither the ability to afford the sense of presence. Recall that immersion is "*what the technology delivers from an objective point of view*" (Bowman and McMahan 2007). The words in the book have to be interpreted by a reader before we can say that an experience has occurred. Since the reader is involved in this context we cannot say that books deliver experiences from an objective point of view. As a consequence of this notion the effect - presence - the sense of being there, cannot be considered a derivative of books even though some people might claim that they have felt so engaged with a book and its characters that they felt they were part of the story. This attachment and emotional relationship is however accounted for by *involvement*.

Inspired by the concepts of *cognitive absorption* and *flow* we draw a relation between the inherent structural elements of games and their possible effects. When a player is either engaged in the gameplay by dealing with the challenges, or emotionally attached to the development of a plot in a story and sympathises with characters, we have involvement. Involvement is as opposed to presence admittedly determined by the individual that engages in the experience. This leads us to talk about the last part of the model - the user and the context seen in the bottom of the model. The chance of achieving involvement depends on the individual's personality traits, tendency to become engaged and internal motivation to become involved.

Between *experience* and *user* we have placed *context.* The curved lines indicate that context is a distorting factor meaning that it will create unique experiences for every game session to individual users. It depends on the motivation to participate, the participant's mood, other participants, and the participant's experience with the given platform and content.

6.3 Preliminary Conclusion

We could distinguish between the two states - presence and involvement - and concentrate on separating them. What we found in the popular descriptions of immersion was covering both aspects. You can feel presence without involvement, in the case of listening to a concert from an impressive sound system without engaging in the genre of music that you hear. By contrast you can feel involvement in the story of a book without the feeling of presence.

We prefer and use the distinction between immersion, inherent structural elements, presence, and involvement in our situation. This definition has some clear advantages compared to other definitions:

- It clearly distinguishes presence from involvement, which is important and not made clear within game immersion research!
- It separates different aspects of an experience, and unlike game immersion it takes the whole system into account by making a distinction between form and content.

7 The Experience Cylinder

In this section, we will explain the Experience Cylinder in greater detail. We will present the origin of the cylinder and draw comparisons to similar installations. After this we will describe the physical structure of the cylinder to be able to highlight problems in the structure that have affected our work. Following this we present our development platform and the most significant tools and applications we will use to develop our game prototypes.

7.1 Introduction

The Experience Cylinder is the technology that drives the practical development involved in this project. The cylinder is, as formerly mentioned, used by Roskilde University to research new experience forms. In collaboration with the Viking Ship Museum a 2D experience was made for the cylinder. The experience involves pictures and video of the reconstructed Viking ship *The Sea Stallion* and its trip from Roskilde to Dublin and back.

The development described in this report however is driven by the possibility of turning the Experience Cylinder into a virtual reality display and use it for 3D virtual environments. This way one could potentially get a feeling of standing inside a computer generated world as the Experience Cylinder resembles a class of system known as a CAVE.

The Experience Cylinder consists of a cylinder shaped canvas hanging from the ceiling. Six projectors are mounted inside as precise as presently possible, shooting on to the opposite side of the canvas, so that it becomes a large 360° display. As mentioned previously, an infrared depth measuring camera - the Microsoft Kinect - is mounted in the ceiling above the centre of the cylinder in order to track the user(s). It is mounted pointing directly into the floor. Outside of the canvas are six loudspeakers, each one positioned behind the centre of a projector image. A four array subwoofer setup is placed outside as well.

As the Experience Cylinder is the physical foundation for our project, we will in this section analyse the different elements of the Experience Cylinder regarding identifying and dealing with the challenges of developing an immersive game prototype.

7.2 Similar Installations

Before we turn to the detailed description of the construction of the Experience Cylinder we here give two examples of similar installations that apply gaming in a virtual reality setting. From these examples we point out a number of similarities and differences.

The game industry is one of the major driving forces for developing virtual reality platforms for entertainment-based applications (Badiqué, et al. 2002).

Due to the cost of large virtual reality installations these kinds of location-based entertainment installations are mostly encountered in theme parks or video arcades and not in private homes.

7.2.1 Disney Quest Example

An example of an interactive virtual reality attraction is the Disney World *Battle for the Buccaneer Gold* (Shochet and Schell 2001). This is a nice example of game development in virtual reality context where platform and game content is thoroughly merged.

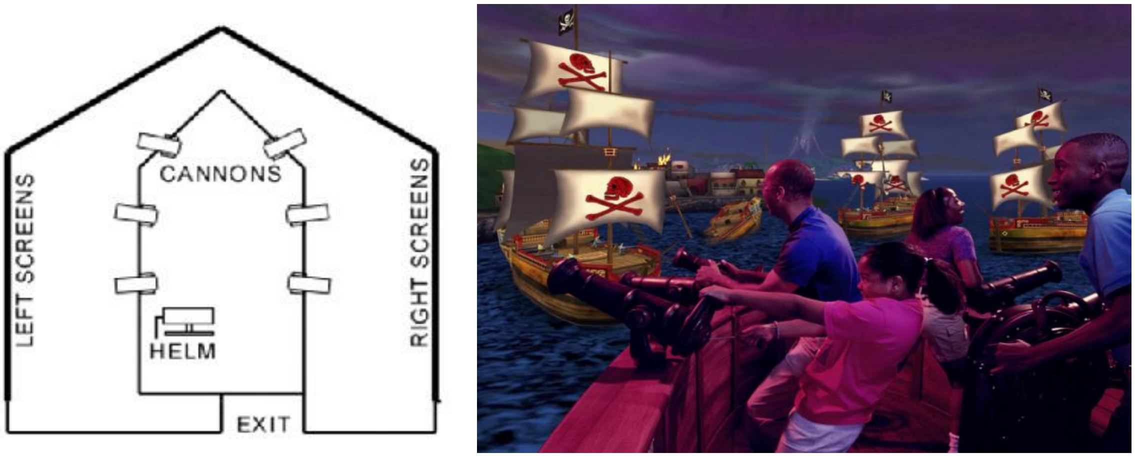


Illustration 7-1: On the left the layout of the LBE-installation and on the right is a photo of the game in action. Rightmost a player is at the rudder and the other players man the cannons. The ships in the background and the water are computer generated imagery.

Disney's *Battle for Buccaneer Gold* is mentioned as an interactive theme park ride. One guest steers and three other guests' man cannons used to defeat virtual enemy pirate ships. The installation uses a so called wrap-around screen with stereoscopic 3D and directional surround sound as well as a motion platform for moving the physical space – a pirate ship.

The description does not make explicit whether the display completely surrounds guests but as the illustration indicates it spans from the floor to the ceiling of the room.

7.2.2 Star Wars Example

Students from Medialogy at AAU - Copenhagen have investigated game development for a traditional CAVE design (Livatino, et al. 2006). They use the Star Wars universe as the narrative foundation for creating a game specifically for a CAVE-like installation. The game idea is taken from the original movies in which the main character must practice his lightsaber skills. The idea is to let the player experience the same kind of practice. As seen in the Illustration 7-2 it appears as though the player holds a lightsaber. The challenge in the game is to deflect laser beams emitted from what is referred to as *the training remote*, see the upper left corner on the screen in the Illustration 7-2. Points are gained when laser beams are deflected and the level of difficulty is increased as more points are gained. The player is wearing 3D glasses and both the glasses and the physical lightsaber hilt are tracked to adjust viewing perspective and the virtual part of the lightsaber according to the player’s movement.

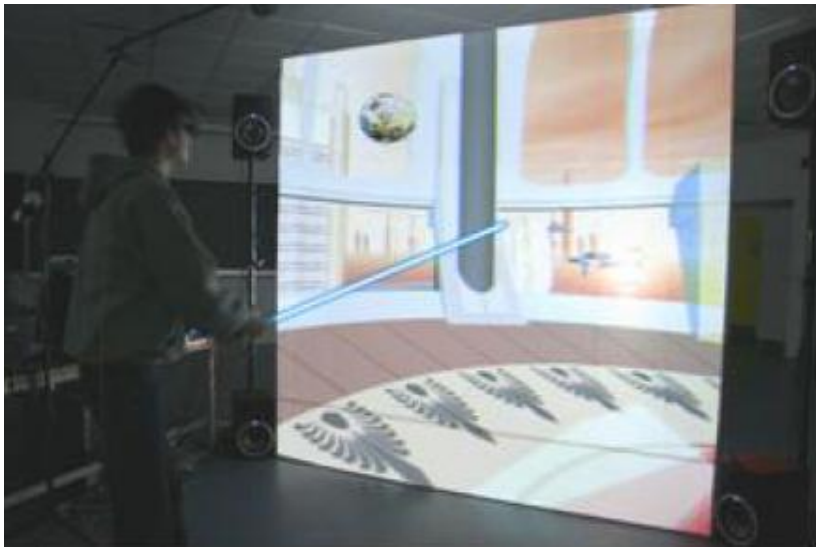


Illustration 7-2: On the display is the game world. The ball in the upper left is the enemy and the light blue line is the virtual part of the lightsaber. The player is holding a physical hilt for the lightsaber and the illusion of holding a lightsaber is created by tracking the movement of the head and the hilt. The player is wearing 3D glasses to allow the illusion that the lightsaber appears in front of the display.

7.2.3 Comparison

What these two examples have in common which is different from our prototype is that they build upon a familiar game world. The Disney example use the mythical version of the pirate world and the other example builds upon the Star Wars universe. This enables players familiar with these worlds to easily put the specific game in a narrative context.

Another commonality between the examples is the use of stereoscopic 3D technology. This technology, as opposed to our platform, gives the opportunity to visualise game elements in the physical space in front of the display.

What is different between the examples, is the control mechanism and the mapping between physical interface and corresponding action in the virtual world. In the Disney example the physical shape of the control mechanism gives an indication of how it is used as it builds on the understanding of how a real cannon works. In the Star Wars example the player is also equipped with a physical controller that is virtually extended into the game world. The 3D effects give the illusion that the blade of the lightsaber begins at the physical controller - the hilt - and ends in the virtual world. This diversity between the examples is distinct but we dare to say that both examples can be classified as a CAVE.

The Experience Cylinder is quite similar to the understanding of what a CAVE is, but is of course different in the way that it is built because of the original intended application for *The Sea Stallion*. It is not shaped like a box but like a cylinder. It has no stereoscopic 3D display technology. It is larger and built for more than one person and is also fairly cheap compared to other facilities, see (Andreasen, et al. 2011), (Juarez, Schonenberg and Bartneck 2010), and (Livatino, et al. 2006).

The following sections describe the setup as it was when this project was initialized.

7.3 Construction and Technologies

We consider the physical construction of the Experience Cylinder. Built from inexpensive materials, it makes a good start or prototype for investigating content on large displays.

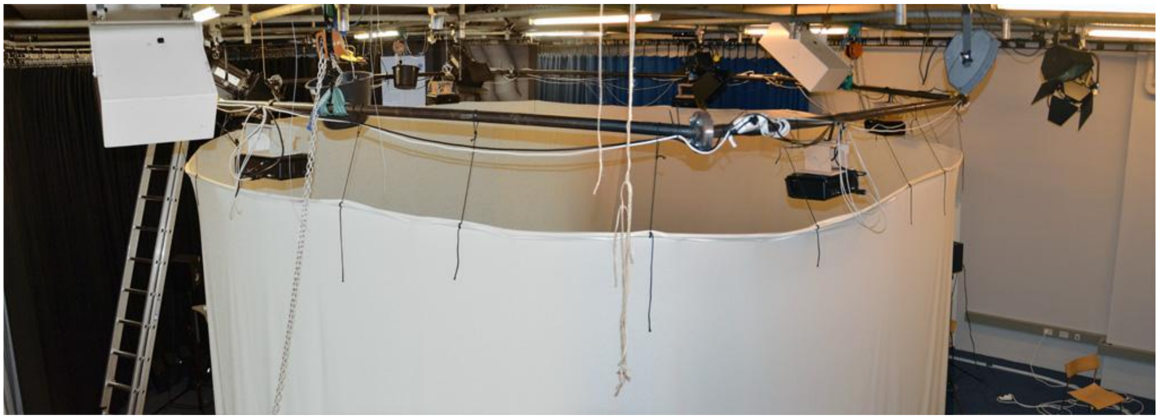


Illustration 7-3: A full view of the Experience Cylinder showing the general construction. Notice the black projectors, the rig of steel tubing, the attachment to the ceiling by the three orange pulleys, and the white canvas and its shape.

7.3.1 Rigging

The rig for the canvas and projectors is a regular hexagon with sides of approximately 300 cm, created by ⌀ 6 cm steel tubing. It is mounted to the ceiling in a 3-point lift system. The three points are placed evenly each in the centre of three of the sides in the hexagon.

7.3.2 Display Surface

The display surface is a large piece of canvas attached to ⌀ 1 cm plastic tubing in a circle of approximately 600 cm in diameter. The plastic tubing and canvas is mounted via strings to the hexagon. Notice on Illustration 7-3 how most of the strings hang at an angle.

7.3.3 Projectors

The six projectors are mounted so that each is mounted on each side of the hexagon support on the opposite side of where they project to. But because the aforementioned 3-point lift system was attached to the centre of the hexagon sides, it was decided to mount all the projectors displaced from the centre on the hexagon sides while using the projectors' lens shifting to correct for the displacement.

7.3.4 Kinect

A Kinect, which can be used for motion sensing, is mounted on the ceiling in the centre of the Experience Cylinder. This is used for the control for *The Sea Stallion* cylinder software. A traditional mouse and keyboard can also be used for input to the computer of the Experience Cylinder.

7.3.5 Computer

The computer was a Macintosh server with the operating system Mac OS X. It contained two Intel Xeon processors, each with four processor cores. Two ATI Radeon HD 5700-series graphics cards were mounted. Furthermore the computer had 8 Gb of RAM. In order to connect with all six projectors each graphics card had a Matrox TripleHead2Go Analog Edition mounted that connected three projectors to one output.

7.3.6 Sound

An M-Audio ProFire 610 external sound card is attached to the computer. This sound card is different from a more typical sound card in that it delivers up to ten channels that is fully controllable from within the software. The six satellite speakers are connected to this in six of the ten channels in addition to four 10" subwoofers connected to only one channel. All of them are active meaning that they have their own built-in amplifiers. This is considered a 6.1 surround sound setup. This notation is used to indicate the number of channels. The number before the period indicate the number of full-range channels, meaning channels that plays the full spectrum of sound, and the number after the period indicates the number of low range channels.

8 Development Platform

In the previous section we explained the different parts of the Experience Cylinder and what technical opportunities we had to work with at the start of the project. In this section we explain different aspects of making video game content based on the integrated game development framework Unity3D. We do that by explaining on a practical tool based level what is needed for developing games with the Experience Cylinder technologies in mind.

First off we based our decision to use Unity3D on the CaveUT developer's decision to do the same. We knew by their decision that it were possible to render a scene with a similar result to what our proof of concept test of CaveUT had. Since both solutions had that specific capability and it seemed easier to implement new game ideas in an editor rather than to modify an existing game, the choice was easy. Most of the other software solutions were selected for their compatibility with Unity3D. Additional details will be explained later though.

Following is Illustration 8-1 showing the connections between the applications we use and how they are connected to the Experience Cylinder. The majority of them concern software or file types and standards used within the computer, but additionally also the connection between the software solutions and the previously mentioned setup of hardware. The software is shown by their respective logos, and the specific hardware is shown to the right in addition to the 3D model of the Experience Cylinder where applicable. In between, arrows show the direction of workflow that either we ourselves or the software execute. Following the illustration we will explain the parts and how they are used.

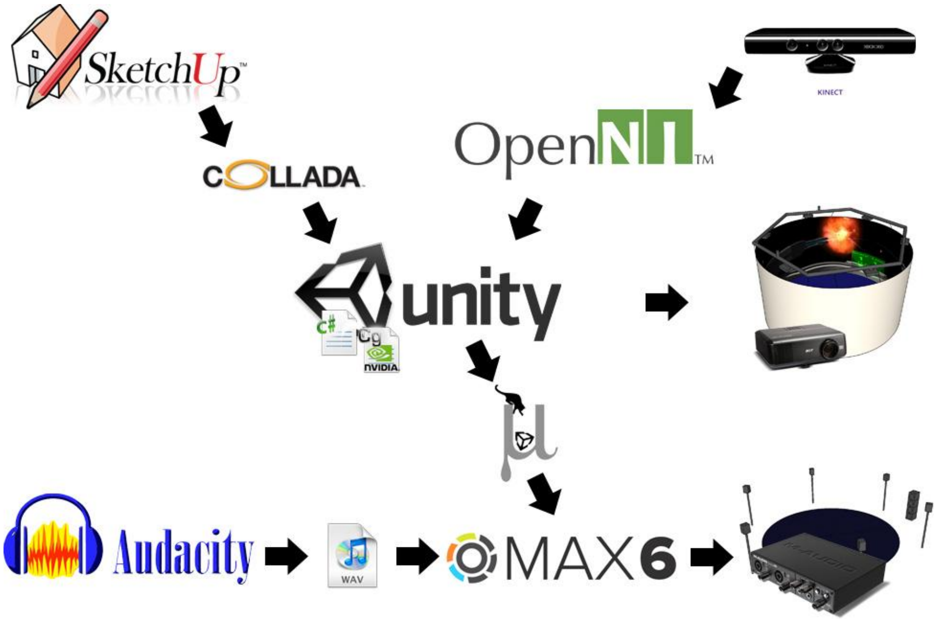


Illustration 8-1: This illustration shows a graphical layout of our development platform. The platform consists of different software solutions. These are illustrated with their individual interdependencies as well as between file types and the Experience Cylinder hardware.

8.1 Unity3D

Unity3D is a game development platform. It runs on Windows and MAC OS X as well as compiling to a wide variety of platforms – PC and Mac but also Xbox, PS3, Nintendo Wii, iPhone and Android. The main parts of Unity3D are the editor and the game engine. The editor is the part used under development and the game engine is the interpreter that generates the graphics based on the graphics API that corresponds to the platform of choice.

The editor organizes the game development into three: *assets, scenes and projects*. Any single game is organized as a project. Scenes are associated to a specific project and can often be seen as the different game levels in a game. Assets are even smaller bits that make up the game. Models, shaders, scripts, lights and sounds are all assets.

The editor interface is divided into a number of panels. For example a panel with a file system hierarchy containing all assets, a game view panel that shows a pre-rendering of the game, a scene view panel that graphically displays the whole 3D world and visible game objects in the scene. Here every game object can be manipulated by using the mouse. In the right side inspector panel we see the settings for a given game object, in a way that makes it easy to configure the individual game element.

Unity3D comes with a built in physics engine, able to handle gravity, movement, collisions etc. It also contains a number of built in presets that allow for easy creation of game objects and functionality that would be very advanced to make from scratch. For example springs, hinges and joints all of which can be configured in a number of ways, for example to break when subjected to more force than they are configured to withstand. These are just a few of the options available in the physics engine.

The scripting engine in Unity3D allows game creators and programmers to create complex behaviours in the game. This is done by writing scripts that create the intended behaviours and then attaching the script to relevant game objects to use the behaviour in the relevant scenes. Three scripting languages are available in Unity3D, C#, UnityScript and Boo. UnityScript is derived from JavaScript. We chose to use C# as it more closely resembled programming languages we have used in earlier projects, and since none of us had any experience with JavaScript or Boo or C# for that matter.

Unity3D also supports shaders, which are programs running on the graphical processing unit. Shaders are used to calculate rendering effects and can in Unity3D be programmed in the multi-platform language CG, short for C for Graphics.

8.2 OpenNI

OpenNI is a framework, which has been developed with the intention of creating a common way to interface between programs that require 3D sensor data, algorithms that process these data, and the sensors that produce the actual data. Program developers can make programs that use 3D sensor data without limiting themselves to specific sensors, hardware producers can make new sensors that still work with existing programs, and new algorithms can be written to process sensor data without affecting program functionality or being dependent on specific sensors (PrimeSense, et al. 2010).

OpenNI also includes several middleware modules, which can process the raw data from the Kinect. These middleware modules include ways to display the depth camera view graphically, modules which can keep track of several different users in the camera field of view and modules to generate a sort of radar view to visualize where different users are located in relation to the Kinect and each other. Perhaps more importantly, there are also modules to provide a skeletal tracking feature for a user, which is needed for more advanced gesture control and movement tracking.

OpenNI is coded in C, but we can write scripts in C# because the package connecting the OpenNI framework to Unity is configured to handle all connections to OpenNI, which allows us to write all our Unity code in C#, one of the supported scripting languages in Unity.

We chose OpenNI because other people had already developed a package that connected Unity and OpenNI, and because OpenNI appeared to be frequently updated and well documented.

8.3 SketchUp

In order to generate any visible objects for a 3D virtual environment one has to draw them in a 3D modelling program. Unity3D supports simple modelling but it is not versatile enough. SketchUp is a modelling tool and comes in a free edition (Trimble n.d.). It has the advantages of being very easy to use in order to get started. Most of the modelling we use in our game prototypes is performed with SketchUp. Additionally the 3D models used within this report were created in SketchUp as well.

In order to use the models from SketchUp within Unity3D we had to export the data to a file format that Unity3D can import. The new free edition of version 8 was able to export in the XML based 3D file format called Collada. The extension of these files is *.dae. We imported these files without trouble as long as we remembered to import the textures first. The textures are the images on the surfaces of the model.

8.4 Max 6

Max is a highly modular, visual programming language for music and multimedia with an API that allows for the development of new 3rd party routines. Originally Max is designed to handle synthesizers and samplers for making electronic music, but a set of audio extensions called Max Signal Processing (MSP) now allows for the manipulation of real-time digital audio. That is why the language is also known as Max/MSP. Max programs are called patches and made by connecting building blocks of objects. These objects are themselves programs that either receive or transmit input/output to other objects via inlets and outlets.

8.5 Audacity

Audacity is an audio editor software application that we can use to manipulate pieces of sound. It features audio recording, post processing, and mixing of audio. It is able to edit different audio file types. We chose to use the wave file type as Illustration 8-1 shows. On the website where we got the original sound files both the mp3 and wave file formats were available (Soundbible 2012). We chose the latter because of its ability to deliver higher quality sound than mp3.

9 Choosing a Case

Our specific case was not chosen until after a preliminary literature study and some initial work with different software tools to familiarize ourselves with the Experience Cylinder and the kinds of software we needed to display a virtual 3D environment in the cylinder. We knew from the start that we wanted to create a video game for the cylinder, but the specific genre of our video game prototype had not been decided yet. Thus, part of our initial investigations included coming up with game ideas that utilised the different technological elements in the Experience Cylinder as much as possible without sacrificing the overall game experience.

Display .... Among the technological aspects of the experience cylinder, we considered the 360° panoramic display the most important aspect and chose it as our main focus area. An experience with CaveUT was that even though the player's were able to see 360° of the virtual environment, the player primarily utilised the main display while playing. On contrary we wanted to design our game prototype to utilise the entire screen area in a meaningful way.

Controller .... Since the Kinect were mounted already, we wanted to use this as the means to be able to control our game. We wanted the player to be able to aim and point in the game only through his own body movement, rather than using a conventional controller like a computer mouse. If it was at all possible we also wanted a way for the player to move around in the virtual environment, only through the use of body movement within the cylinder.

Movement .... Movement needs to be considered in combination with the use of the display. The primary reason for the previously mentioned issue of only using the main display in CaveUT is the game design of Unreal Tournament. In order to get the full potential of CAVE-like systems you need to design the game to come as close to real world aspects as possible. One central aspect of the real world is that the environment around you is fixated while you move or look around. The opposite happens in CaveUT where the virtual environment moves around while the player is fixated. We did not expect to solve this within navigation which would require additional hardware, but we aimed for a fixation of the virtual environment rotation wise. Since the display is all around you can just turn your head or body like in the real world.

Sound system .... We wanted to utilise the cylinder's sound system, as a way of cueing the player in on aspects of the game that were potentially out of his field of view. This also is a way of creating sudden shock in the player to indicate immediate danger or failure. Sound is also very important in consideration to providing a sense of space in addition to the display and the 3D virtual environment.

Besides the technological aspects of the Experience Cylinder, we also had to make decisions regarding the relation between the individual game ideas and different aspects of games. Rather than just coming up with game ideas, we decided to look at the individual aspects and judge their relevance to our project. Both in relation to what we felt we wanted to investigate in our game experience, but also what we deemed achievable within the time limit of our project.

Narrative............. Writing a story to support a narrative was quickly excluded, solely on the basis of the scope of our project. it was unrealistic for us to come up with a narrative element because none of us had any experience in creating a narrative.

Gameplay............ A focus on the player's reaction time and coordination seemed more immediate and appropriate for the Experience Cylinder. Our game prototype could be based on the player's reaction time by presenting objects on many different parts of the display and simulate sounds coming from many different directions. If we simultaneously utilised the Kinect as the controller for the game, the game could challenge the coordination and reaction time of the player through his or her physical movement. On the other hand designing complex puzzles and challenges requiring logical thinking skills would require extensive work from our part and we deemed unfeasible to implement within the scope of the project.

After considering these aspects of games, we had some considerations about how the immersive properties of the Experience Cylinder could augment our choice of gameplay. This included that the game should: utilise the entire screen area of the Experience Cylinder, utilise the physics system in Unity3D to more closely imitate aspects of the real world, and utilise proper directional sound to enhance the user's awareness of the virtual environment.

9.1 Basic Case Concepts

This section provides basic information about plausible game concepts we considered while developing the needed case(s) in regard to the Experience Cylinder.

9.1.1 Rail-Shooter

The concept of a Rail-shooter is that the player's movement around the game environment is partly controlled by the game itself. Most often, the player will be able to aim his gun around freely, but character movement between different parts of the environment is controlled by the computer. One example of this is where the player is moved from room to room. As a player's character enters a room, he has to kill a number of computer generated enemies before he is taken to the next room. The games *Time Crisis* and *Virtua Cop* are examples of this game type. Also an actual rail could exist in the game. This could then be compared to the classic ride at the amusement park where you sit in a carriage with a gun mounted and travel along the rail from start to finish, while you shoot at moving targets.

A game prototype in this genre could utilise the entire screen area by designing each scene, so that enemies appear all around the player and the player has to be aware of many parts of the screen. The directional sound system could be used to indicate enemy locations and to attempt to distract the player.

9.1.2 Base Defence

In a base defence game the player controls one or more defensive towers and has to defend himself or points of interest in the game from approaching enemies. The idea is based on the game Missile Command, in which the player has control of a few defensive towers with which to defend bases from incoming enemies. In games of this type, the defensive towers are usually viewed from outside the tower, to give a sense of control and overview.

The Experience Cylinder presented a unique opportunity to put the player into the defensive tower itself, letting him experience the game from a first person point of view, to give a more intense game experience. The sound system could be used to indicate incoming enemies or used to create attack sounds of varying intensity depending on the distance of the sound origin or the severity to the player.

9.1.3 Virtual Sightseeing

This could be considered a concept closer to the simulation genre. It does not have to be a game with any other goal than travelling around seeing the virtual environment in question. This concept could opposed to the other concepts involve free movement in the virtual environment as the central game structure. Additionally in combination with the Experience Cylinder, this concept could apply the possibility of realizing a *fixed world - turning player relation*.

In a virtual sightseeing prototype the display could show the surrounding environment and the sound system could provide ambient sounds to create atmosphere.

9.2 Choice of Case

The Base Defence concept was the most simple to start our investigations by. A finished Base Defence game could be reused by adding the Rail-Shooter features of transport to varying locations in order to investigate movement in virtual environment. So the Base Defence concept was the choice of the first two.

During the project we realized that converting the Base Defence game prototype to a rail-shooter was unfeasible within the scope of the project. So we decided to investigate a separate solution which led to the *virtual sightseeing* concept.

10 The Development Challenges

This section provides an overview of how we have developed the game prototypes. On the basis of our proof-of-concept study with CaveUT and during the practical development of the prototypes, we identified three of four main challenges: *display*, *control*, and *sound*. The fourth challenge relates to the challenge of developing game content. Each of these challenges is presented with a background section and a section that describes our solution.

Displaying a 3D World on a 360 Display... In CaveUT, the player's focus is primarily in the same direction. We wished to better utilise the display by encouraging the player himself to turn and look in all directions. The challenge we face here, is to represent the 3D world in a meaningful and realistic way on the 360° display, while the game design is based on utilizing the entire screen area.

Control.. Traditional control with mouse and keyboard restricts player movement. We want the player to be able to move around freely. The Experience Cylinder is equipped with an infrared camera and we want to utilise it as the controller for the game. The challenge is to make proper use of the features of the Kinect.

Directional Sound.................................. We did not get any experience with audio from our proof-of-concept study, and the hardware for sound capabilities in the Experience Cylinder is previously only used with a specific technology. We face a challenge in playing sound from one sound interface into our game development environment.

Game Elements..................................... The last challenge deals with the making of content for our game. Unlike the proof-of-concept study, we ourselves had to create and invent the game elements central to the game concept. We needed relevant game elements to move and behave in a realistic and predictable manner as well as portray a visual expression reflecting their expected purpose or behaviour.

10.1 Displaying a 3D world on a 360° display

This section is concerned with the problems and solutions about how to show a 3D virtual world on the inside of a 360° panoramic surface in order to create the experience of standing inside a virtual world looking towards the horizon in all directions. The pre study involving CaveUT showed us the proof of concept and in the following we show how we solved similar issues on our platform.

10.1.1 Platform Issues

First off the Mac OS had no support for AMD CrossFireX or AMD Eyefinity. AMD CrossFireX creates a data bridge between the two GPUs in order to combine their computing power. AMD Eyefinity combines multiple displays in a way that makes the OS see them as one. The lack of support for AMD Eyefinity meant that in Mac OS we had two displays containing each of three projectors with no ability to make the OS see all the screens as one. Each projector was fed with a resolution of 1024 x 768 which means that each of the two displays in the OS had a total resolution of 3072 x 768. On top of that we found that Unity is only able to show a full screen rendering on one monitor. So we were only able to get fullscreen on the one 3072 x 768 display which corresponded to only 180° of the Experience Cylinder.

10.1.2 Display Surface Limitations

As formerly mentioned, attaching the plastic tubing ring to the hexagonal shape directly with strings cause problems. Because of that and the fact that the plastic tubing is not rigid enough to keep its cylindrical shape, the supposedly cylindrical display is distorted towards the more rigid hexagonal shape of the support, see Illustration 7-3 on page 39. This means that instead of a perfect circle we really have an at least much softened hexagonal shape as the display. Additionally, the canvas has a tendency to hang unevenly which creates arbitrary folds and curves. These conditions distort the image that is displayed on the canvas.

10.1.3 Projectors and Placement

The projectors presently mounted in the Experience Cylinder are not capable of curved-surface projection correction and therefore are not able to correct for the fact that they are used to project on the curved surface of the cylinder, rather than a plane surface. The following Illustration 10-1 shows two images: the left image shows only the cylinder and a red, transparent, and fictive plane where one of the projectors is supposed to project to; the right image show the fictive plane again but this time combined with a projection outline of where the projection will hit the plane and the canvas.

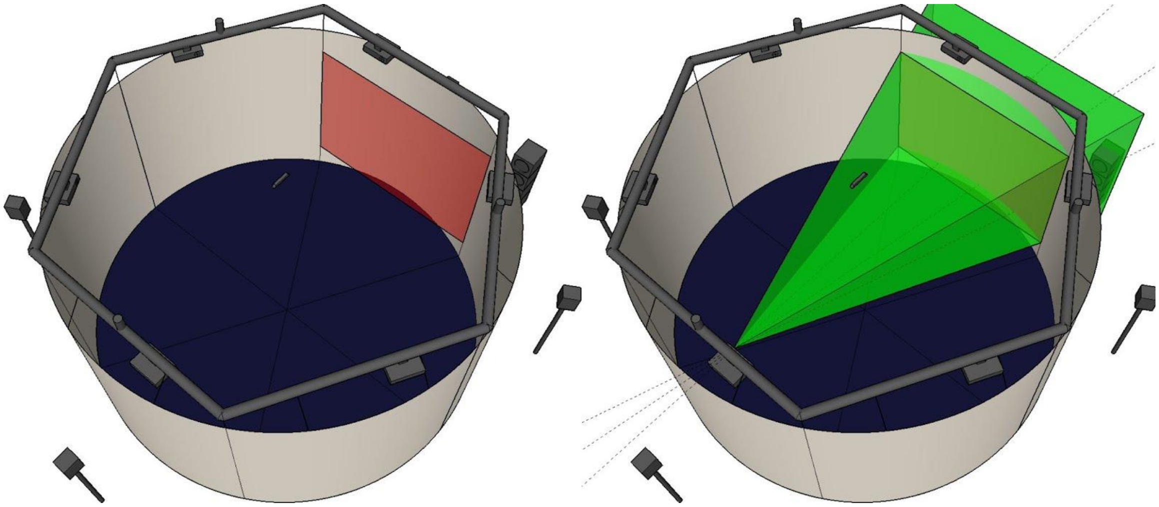


Illustration 10-1: The image to the right in this illustration shows an outline of the light from one projector. The light from this projector is supposed to hit a plane surface which is shown as well. The fictive plane surface is shown in red isolated in the image to the left. Since the Experience Cylinder is circular the light continues its travel of direction from the fictive plane to the actual surface. This results in distortion of the image.

This has the effect of the image becoming distorted. What happens is that from beyond the fictive projection plane the projection light continues its travel in the originating direction and therefore places the pixels in spots where they were not intended. The following Illustration 10-2 shows the fictive projection plane directly from the centre front, together with the projection outline as the right image in Illustration 10-1. It is clear that the distortion result, in case of a perfect circular surface is a curve as well. It also shows that the distortion grows larger from top to bottom and that it curves downwards. This is a result of the high projector position above the display. For comparison, a projector projecting directly from the vertical centre of the display, would result in curves that at the top would curve upwards and at the bottom would curve downwards. We define this distortion as a vertical pixel displacement.

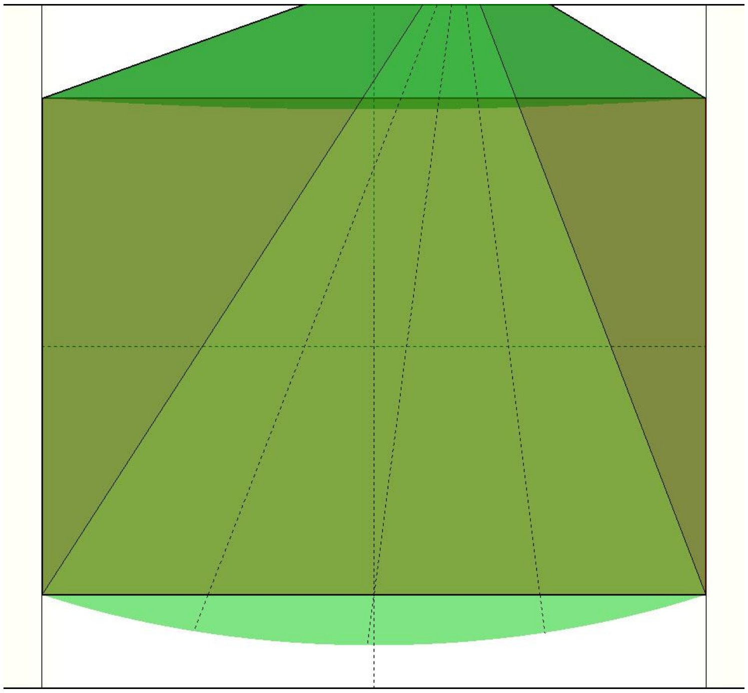


Illustration 10-2: This illustration shows the distortion of the present short throw projectors. The image is an orthogonal view of our 3D model of the Experience Cylinder. It is viewed directly from the centre of the surface where one of the six projectors displays.

Furthermore Illustration 10-2 above shows an additional pixel displacement. The curved distortion combined with a projector that is not centred, makes the pixel distortion problem worse. The illustration shows three dashed lines that originates from the centre and ends at the bottom curve. The middle dashed line represents a pixel that is intended to be displayed in the exact centre in the bottom of the display. With a centred projector this pixel would hit in the exact spot of the turning point of the curve, but as the illustration shows, it does not. It is shifted slightly to the left where the dashed line disappears through the cylindrical surface.

A centred projector would still result in those shifts in the right and left of the three dashed lines, but they would be equally distributed in left and right direction and not at different angles as the current projector setup. We define this distortion as a horizontal pixel displacement. This distortion is not as distinct as the vertical distortion. Actually we did not notice this displacement with the current projector position until quite late in the course of the project.

Also, by deciding to position the projectors on the opposite side of the projection surface, a problem occurs: a person of normal height inside the cylinder obstructs the projector light and either creates a shadow on the display or in some positions gets blinded by light. To overcome this, short throw projectors have already been bought and have been considered in both a rear projection and a front projection setup. A quick test showed that rear projection was not possible with the current canvas because too much light passes through and therefore blinds the user.

The short throw, front projection setup will have other issues. The following Illustration 10-3 shows the 3D view and the result of the curved distortion from such a projector mounting. As the illustration shows, the curved distortion is greater, especially at the bottom.

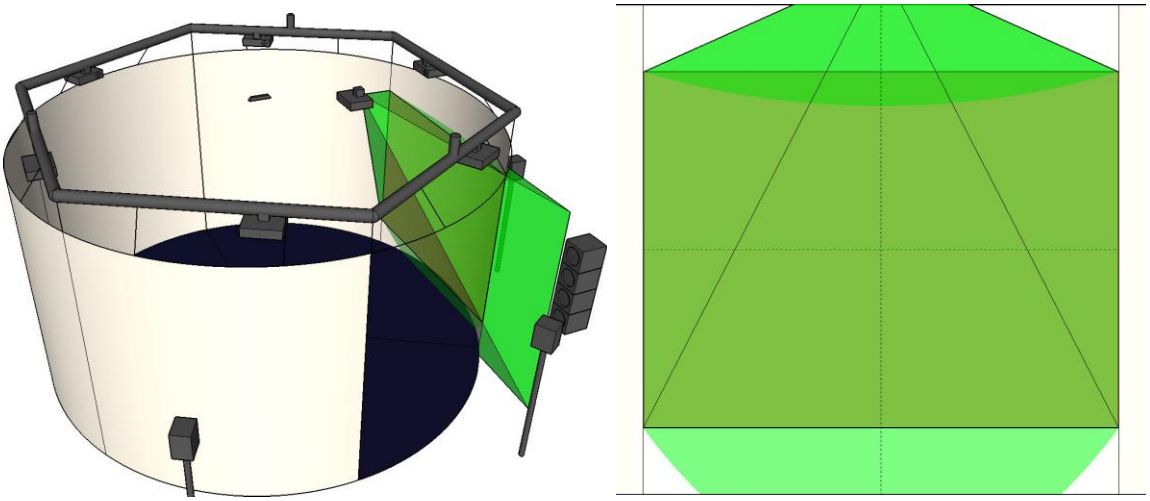


Illustration 10-3: This illustration shows the distortion of a short throw projector mounted in the Experience Cylinder. The distortion gets worse at shorter distances from projector to the display surface.

10.1.4 Virtual Camera in Unity3D

The virtual camera is really a virtual representation of what you would consider a camera that films the world while sending the filmed material to a display. But of course it is a programmed virtual representation of it. Typically games only use one virtual camera for the view and maybe another for a game map. In Unity the virtual camera is utilised by inserting a camera game object into the scene of your application. There is no limit to the number of cameras one can use in a scene.

The camera translates the virtual environment to the user’s screen (Unity3D 2011). Behind the scenes, this is where the mathematical translation of the vertices in the 3D virtual environment into the 2D coordinate set of the user’s screen is introduced.

The virtual camera’s visible area is comprised by the *view frustum* which is the container of the space that could be visible from the camera's point of view, but within the *clipping planes*; near view plane and far view plane. The two planes cut away the potentially visible parts of the scene which are in front of the near view plane and behind the far view plane.

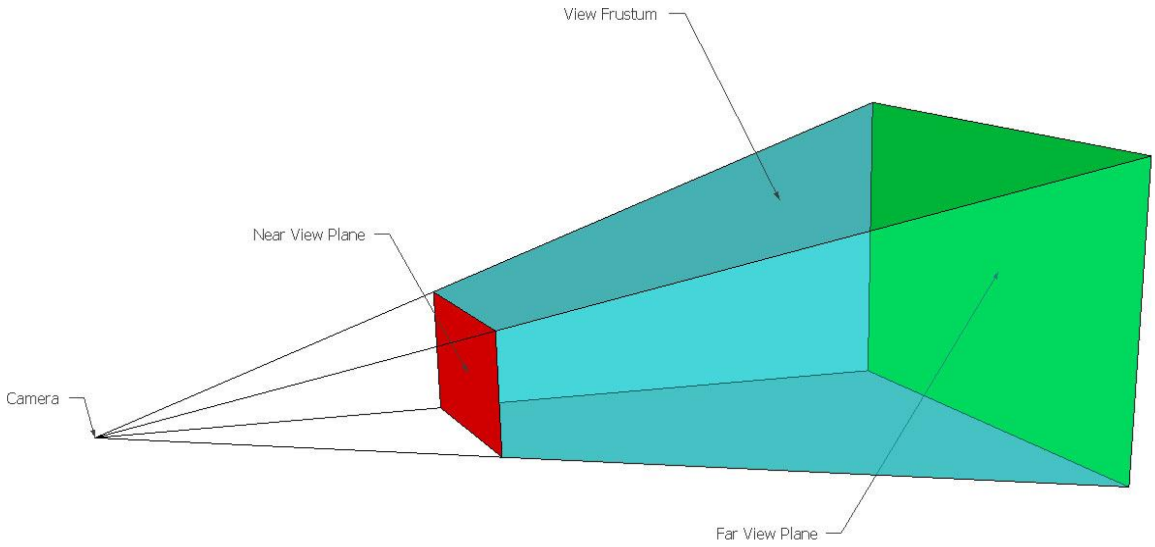
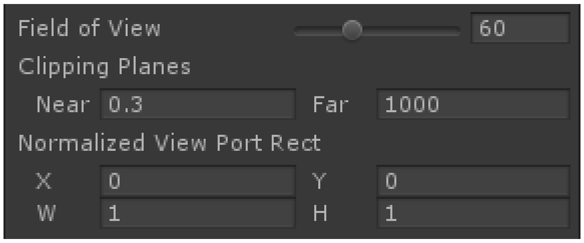


Illustration 10-4: This is a model of a view frustum. It illustrates what is “seen” from a virtual camera, which is everything within the coloured section. The near view plane indicates that everything between the camera and the plane is not rendered. The far view plane indicates that everything beyond is not rendered.

The camera object consists of a series of settings to affect the end result. These settings include, among many others, the ability to set how much of the scene it will render, a setting which is called *Field Of View*, and the rectangular area of the user’s screen where the specific camera will render, a setting called *Normalized View Port Rect* (Unity3D 2011).

The two mentioned settings are the ones we later utilise to create the camera setup we need for The Experience Cylinder.



Field of View 60

Clipping Planes

Near 0.3 Far 1000

Normalized View Port Rect

X 0 Y 0

W 1 H 1

Illustration 10-5: This is a screenshot of some of the camera settings within Unity3D.

○ Field Of View

This is the setting that if changed adjusts the angle of the field of view. If the angle is set to a bigger number the view frustum gets wider both horizontally and vertically. The angle set is the one in vertical direction, but the horizontal angle changes as well in order to keep the aspect ratio between width and height set in the Normalized View Port Rect.

○ Normalized View Port Rect

This setting is adjusted by four values which are the X and Y coordinate of the camera's render position originated at the top, left corner and W (width) and H (height) of the camera's rectangular render area. This feature could be used to create a two player split screen game if both cameras were set to 0.5 in width and the one of the cameras were set to render at the position of 0.5 in X. Then these two cameras would be rendered side by side on the connected display, no matter where the cameras are moved around in the virtual environment.

10.1.5 Background Summary

The issue we have to focus on when creating the ability to view all 360° of the virtual environment is how to properly utilise the virtual camera features of Unity3D. Additionally the issues of the whole experience cylinder construction are a matter that could, if solved, result in benefits to our created experience. The following sections will show the solutions we created and implemented to solve the discussed problems according to the mentioned background knowledge.

10.1.6 Platform

In order to utilise the full 360° of the Experience Cylinder we needed to get outputs to both of the Matrox TripleHead2Go units from only one GPU, preferably coupled with the other by AMD CrossFireX. So to fix the issues of not being able to use AMD CrossFireX we had Windows installed on the machine as well, while possible to start both Mac OS X and Windows. In Windows, we gained the ability to use the CrossFireX as well as Eyefinity and now had one big display. The downsides were that the specific graphics cards we had only had one DVI output and one Mini DisplayPort. The result was that each projector could only be fed with an 800 x 600 resolution, a total resolution of 4800 x 600.

10.1.7 360° Camera

Since The Experience Cylinder is 360° circular there is a need for 360° rendering of the 3D environment to be displayed in the Experience Cylinder. This can be achieved in Unity by setting up a number of cameras circularly with origin at the same coordinate. These cameras are each supposed to show an equally distributed section of the 3D environment each in a different heading of the 360° as in the following Illustration 10-6 of six view frustums.

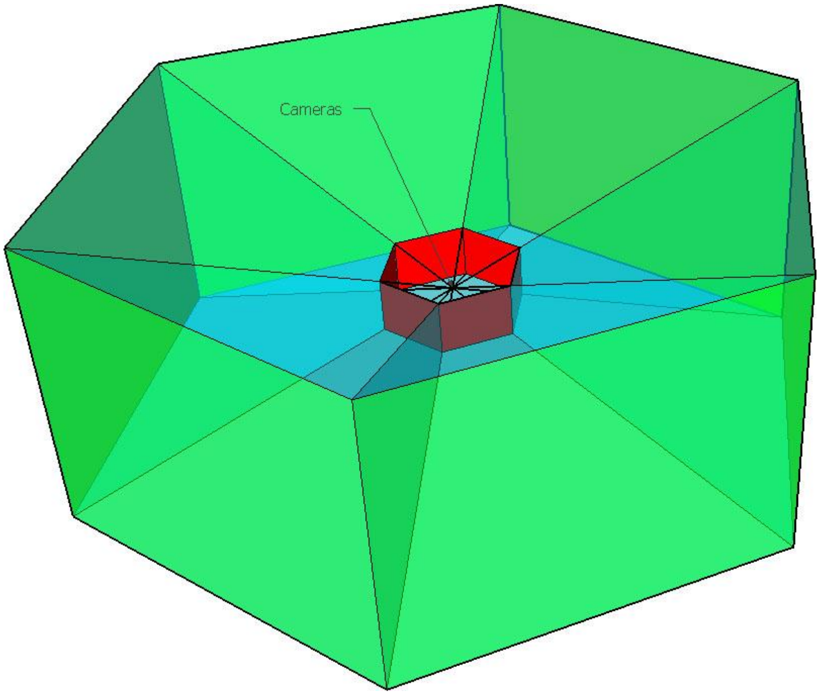


Illustration 10-6: Six cameras added in the same spot, but in six different directions in order to realize a 360° view.

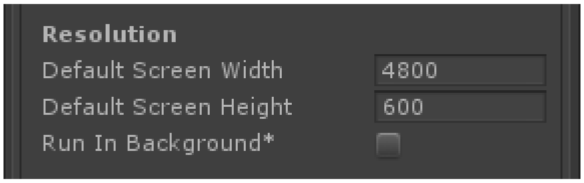
Of course this will result in a regular convex polygon of cameras with as many sides as cameras and not a circle. What happens is that if the 360° camera consists of few cameras, straight lines in the scene will show as if they have bends, where the cameras are convergent. That and the aforementioned curved distortion of the display per projector create clear visualization problems. The preliminary solution to the first problem is that the more cameras we use, the closer we get to a circular shape of the 360° camera, and also a potentially circular view of the rendered scene on the cylindrical display.

The downside is that more cameras in the scene will lower the frame rate of the application. This could in the worst case scenario result in visible lagging. Our small game prototypes were not influenced much, but more comprehensive downloadable Unity test games fitted with our 360° camera were. Our prototypes got great results from a setup with 32 cameras.

To sum up we had two solutions based on the number of cameras used, both with somewhat serious visual and performance issues. To reduce the problem and make it easier to choose between the two solutions we ended up creating a shader program. Thereby we moved some of the work to the GPU. In the shader we introduced a parabolic function that made an approximate fix of the problems by a solution with only six cameras, see section 10.1.80 on page 60.

○ Building the 360° camera

To create the 360° camera some values in Unity needed to be adjusted correctly, which demanded some calculations Unity is not able to supply itself. The first thing to consider is the final resolution, because the cameras will be scaled up or down accordingly. As a case for the calculations, the Experience Cylinder has six projectors, each with a resolution of 800 x 600 pixels making a combined screen of 4800 x 600 pixels. The final resolution is set in player settings reached at the build settings dialog.



Resolution

Default Screen Width	4800
Default Screen Height	600
Run In Background*	□

Illustration 10-7: A screenshot of the resolution settings within Unity3D.

The number of cameras in a 360° view is independent of that, but in the end we chose the same number as projectors for reasons explained later and will therefore show the calculations needed for six cameras.

As mentioned before in order to get the cameras aligned side by side on the display when rendered, the *Normalized View Port Rect* needs to be set correctly. The height of each camera is supposed to be 1, which is 100%. The width is calculated by dividing 1 by the number of cameras. If by the six cameras in question the result would be 0.166667. By having each camera adjusted to the values 0.166667 in width and 1 in height they each render to the total 100% height by 600 pixels and 16.6667% of the total width of 4800 pixels, which are 800 pixels. In this calculation example each camera happens to represent the exact same resolution as each projector of 800 x 600. But that would not be the case if the number of cameras was not the same as the number of projectors.

While all cameras need to have the same values in W and H in the *Normalized View Port Rect* the X value of the position coordinate needs to change by each camera. The first camera needs to be positioned in the coordinate (0,0) and the next two in (0,0.166667) and (0,0.333334), and so on. The cameras will then be displayed in equal sizes side by side on the long rectangular screen of the 4800 x 600 pixels as in the following model of the result.

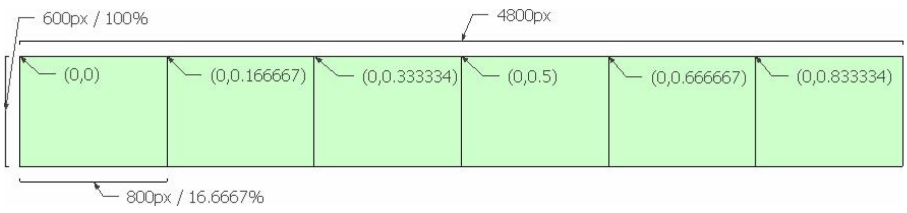


Illustration 10-8: A model of how the six screens of our 360° camera is displayed by Unity3D. The model includes the values needed within the applicable settings.

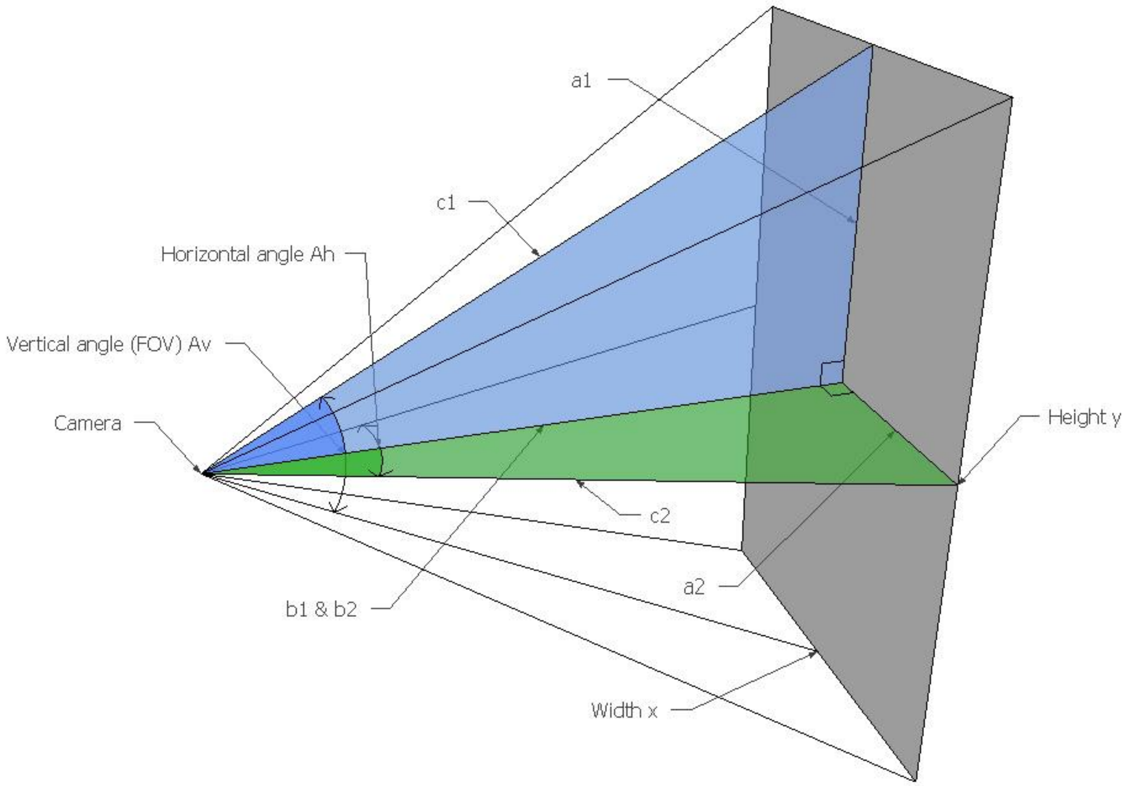


Illustration 10-9: This is a model of a view frustum that contains two triangles necessary for calculation of the Field of View angle. This value is needed in order to create the 360° camera.

The next value to consider, for all cameras, is the *Field Of View*. In Unity it is the vertical angle of the cameras viewing angle. In Illustration 10-9 it is called A_v. When *Field Of View* is changed, the horizontal angle A_h as mentioned changes as well in order to keep the aspect ratio. This is why we need to calculate the two angles based on a plane in the view frustum. A plane available to us is the one defined by the final screen resolution which is 4800 x 600 pixels. As mentioned earlier the height of each camera is 600 pixels. The width can be calculated as 4800 divided by the number of cameras to which the result is 800 pixels. The x and y values are then 800 x 600 pixels.

The horizontal angle A_h is also easy to calculate, again by dividing the 360° by the number of cameras. The result is 60°. So now we have some of the values shown in Illustration 10-9 and need only to calculate A_v, the *Field Of View*. In the green right angled triangle the angle A_2 is A_h divided by two and the side a_2 is the width of the plane divided by two. The value b_2 is not available, but we know it is equal to b_1 in the blue triangle. In the blue triangle we only know a_1 which is the height of the plane divided by two. These values are enough to isolate *Field Of View* by first isolating b in the two triangles and then equating the two expressions shown in the following example.

$$tan\left(\frac{A_v}{2}\right) = \frac{a_1}{b_1}$$
$$tan\left(\frac{A_v}{2}\right) b_1 = a_1$$
$$b_1 = \frac{a_1}{tan\left(\frac{A_v}{2}\right)}$$

$$tan\left(\frac{A_h}{2}\right) = \frac{a_2}{b_2}$$
$$tan\left(\frac{A_h}{2}\right) b_2 = a_2$$
$$b_2 = \frac{a_2}{tan\left(\frac{A_h}{2}\right)}$$

$$\frac{a_2}{tan\left(\frac{A_h}{2}\right)} = \frac{a_1}{tan\left(\frac{A_v}{2}\right)}$$

$$a_2\, tan\left(\frac{A_v}{2}\right) = a_1\, tan\left(\frac{A_h}{2}\right)$$
$$tan\left(\frac{A_v}{2}\right) = \frac{a_1\, tan\left(\frac{A_h}{2}\right)}{a_2}$$
$$\frac{A_v}{2} = tan^{-1}\left(\frac{a_1\, tan\left(\frac{A_h}{2}\right)}{a_2}\right)$$
$$A_v = tan^{-1}\left(\frac{a_1\, tan\left(\frac{A_h}{2}\right)}{a_2}\right)2$$

This formula for A_v can then be utilised to calculate the field of view for the Unity3D cameras. Our example as mentioned has six cameras of 60° with a view plane of 800 x 600 pixels. As a basis according to the Illustration 10-9 by dividing the sides of the view plane by 2, the formula can be used like this:

$$A_v = tan^{-1}\left(\frac{300\, tan\frac{60}{2}}{400}\right)2$$
$$A_v = 46.82645°$$

Mathematically though the important relationship in this formula is the factor between a_1 and a_2. This means that using the original values of 800x600 and not 400x300 would lead to the same result.

The *Field Of View* for each of the six cameras should then be set to 46.82645° and each camera needs to be turned towards the correct heading. Since the cameras are 60° in the horizontal angle, there should be 60° between them. In order to have the centre of the cameras towards the centre of the display it is useful to start at -150°, then the next camera rotated -90°, and so on until we reach +150° for the last camera. In Unity3D, the result is a camera setup that can be implemented in any common 3D world in Unity3D, when attached to the main camera.

10.1.8 Virtual Optical Filter

In order to fix all of the distortions created by the physical setup as described we took the first small steps to an advanced shader solution based on a downloadable Unity3D shader, which initially was created for a different purpose. The ability of the shader that we managed to achieve within the project scope was to create an approximate fix of the most distinct distortion. That is the curved vertical distortion that rises from the combination of projector placement and the curved display surface as described earlier. The following Illustration 10-10 below shows the actual distortion marked with red lines. This distortion can be dimmed by introducing a congruent curved shifting of the pixels in the final image. This will as marked in the illustration lift the pixels from the positions of the red lines up to the horizontal lines that define the rectangular screen surface and the centre.

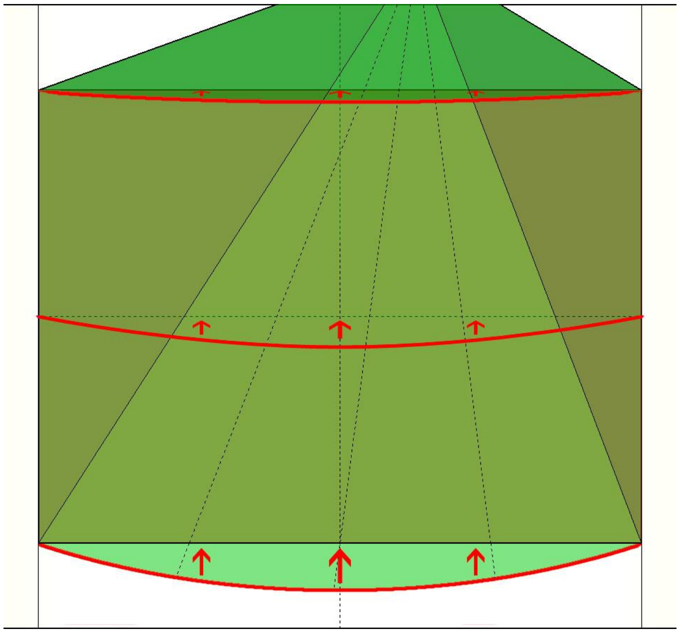


Illustration 10-10: This is a repetition of Illustration 10-2, but this time with additional red curves and arrows that shows the needed shift up of the pixels in order to solve the distortion.

In the shader we inserted the formula for an approximated congruent parabolic function. At first we used one approximated average coefficient that corresponds to the red parabola in the middle. The result was improved overall but was over-adjusting in the top and under-adjusting in the bottom and only somewhat accurate in the middle. Applying a change in the coefficient based on the value of y made the parabolic curve change according to Illustration 10-10 above.

A more precise solution as created in our 3D model would look like the one shown in the following Illustration 10-11. Notice there is an opposite displacement in the light from the projector and therefore no distortion on the projection surface in the final result, as seen in the third image.

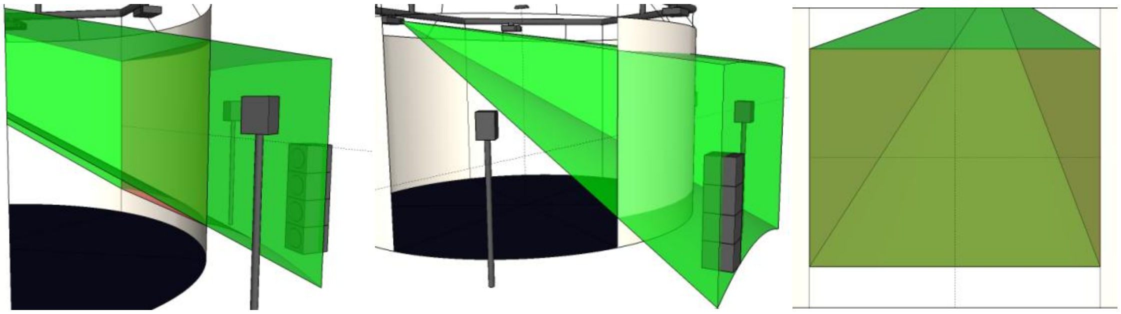


Illustration 10-11: This model shows what would happen if the light from the projector were altered by the correct curves. Notice, that the fictive red plane is visible in the first image, that the upwards curve is distinct in the second image, and that the orthogonal view from inside the cylinder now is without distortion.

To realize an even more precise solution we need to be able to shift each pixel upwards the amount we are able to measure in the 3D model. In order to do that in a more advanced shader a formula for the calculation of that exact value is key and not just an approximated parabolic function. By a number of simple trigonometric operations we found this to be possible. The following Illustration 10-12 shows the shapes and variables needed. The shapes include three right triangles of the colours turquoise, blue, green, one purple oblique triangle, the circle that defines the 360° canvas and the red display surface where the projector is supposed to shoot. What the shader needs to be able to do is calculate the *Vertical Pixel Shift* as defined in Illustration 10-12.

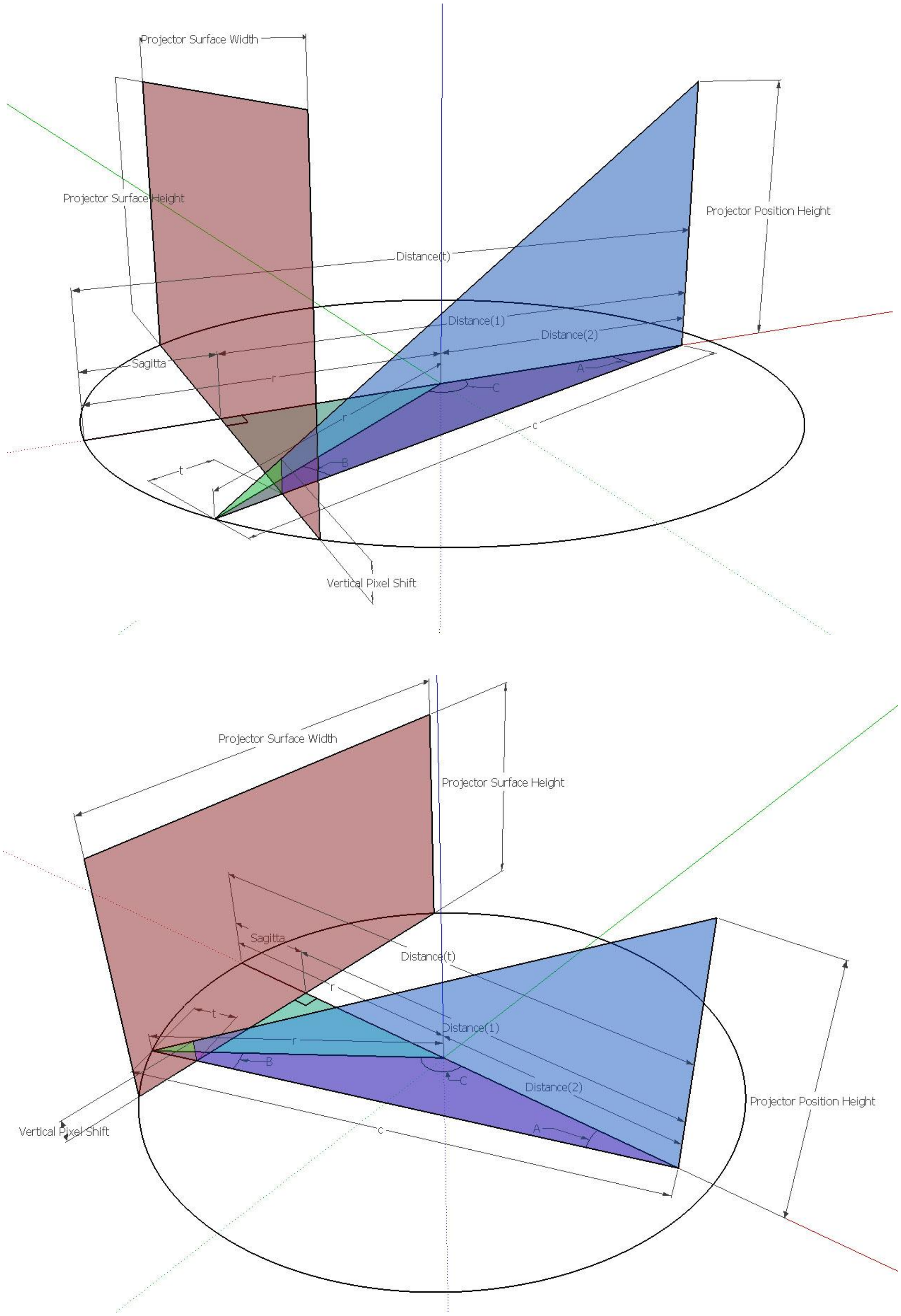


Illustration 10-12: These models show the values and trigonometric shapes needed in order to calculate the pixel shifts needed to resolve the distortion issue. These calculations could be used in a more precise shader program.

The values we know is the projector's position, which is the distance from the canvas , *Distance*$_t$ and the distance from the bottom of the projector surface, *Projector Position Height*, The resolution and number of the projectors, and finally the radius of the cylinder.

These numbers, apart from the resolution which is in pixels, are measured in length units. In order for a possible fragment shader to do the calculations based on a pixel/fragment at hand, all the mentioned variables in Illustration 10-12 needs to be converted to pixels as a unit of measurement. This is the first task of a possible shader program. It is done by calculating the length of the sides of the n-polygon defined by the number of projectors and radius of the circle. A polygon side equals the resolution width of the projector surface and a factor between them can be found which can be used for the following conversions. The calculation of the sides and factor is done as follows:

$$\frac{\sin\left(\frac{180}{Number\ of\ projectors}\right) * 2 * Radius}{Resolution\ width}$$

Furthermore a set of constants can be calculated so the potential shader only needs to do it once. The formulas that follow show some of these calculations. The *Display Top* value is needed for the *Vertical Reference Value*, this way the calculations will work even though the projector would be placed below the height of the *Display Top*.

$$Display\ Top = Projector\ Position\ Height - Projector\ Resolution\ Height$$

The *Sagitta* in this case is the distance from the centre of and perpendicular to the chord that defines the *Projector Surface Width*. In geometry, sagitta is defined as the depth of an arch. This is calculated by the following formula and used to calculate distance 1. Distance 2 is calculated by the radius.

$$Sagitta = r - \sqrt{r^2 - \left(\frac{Projector\ Resolution\ Width}{2}\right)^2}$$

$$Distance_1 = \ Distance_t - Sagitta$$

$$Distance_2 = \ Distance_t - r$$

The rest of the calculations need to be done for each pixel and this is what our shader does in parallel. First off it needs to calculate what we call the *Horizontal-* and *Vertical Reference Values*. They both represent an adjusted value based on the coordinate of the pixel at hand. The *Horizontal Reference Value* goes from half the resolution to zero and back plus one, along all of the x values. We need to add one in order to avoid a division by zero error when calculating pixels in the horizontal middle of the screen. The vertical pixel shift needed in our case is symmetric around the centre of the display. The *Vertical Reference Value* is adjusted accordingly to the display top.

$$Horizontal\ Reference\ Value = \left| x - \left(\frac{Projector\ Resolution\ Width}{2}\right) \right| + 1$$

$$Vertical\ Reference\ Value = y + Display\ Top$$

The following formulas contain all the steps needed to calculate the final *Vertical Pixel Shift*. At first we find all of the angles in the purple oblique triangle, and then we calculate |c| in order to calculate |t| by the turquoise right triangle. Then |t| is used to calculate the *Vertical Pixel Shift*.

$$A = \tan^{-1}\left(\frac{Horizontal\ Reference\ Value}{Distance_1}\right)$$

$$B = \sin^{-1}\left(\sin(A) * \frac{Distance_2}{r}\right)$$

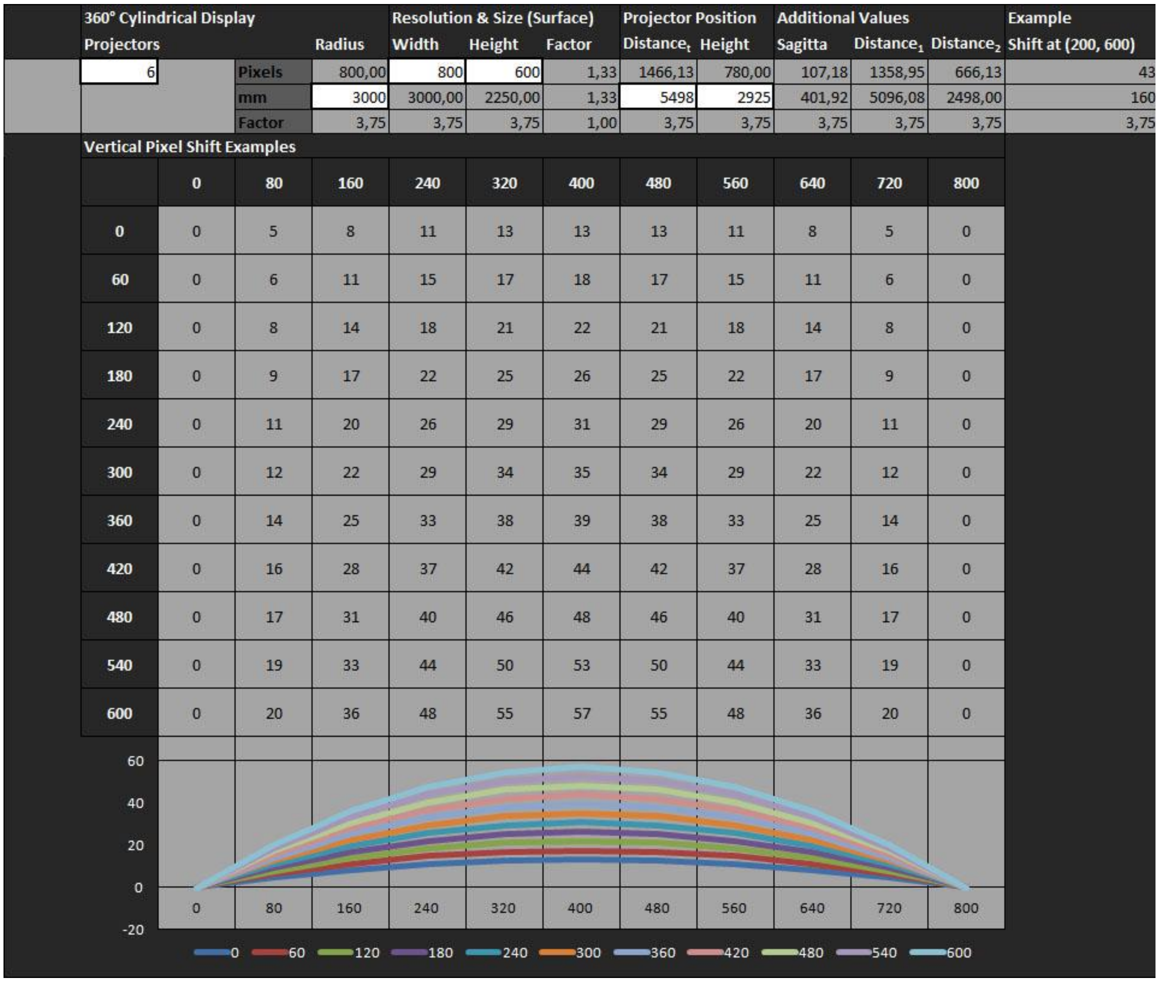
$$C = 180 - B - A$$

$$\mathrm{c} = \sin(C) * \frac{Distance_2}{\sin(B)}$$

$$t = c - \sqrt{(Distance_1)^2 + (Horizontal\ Reference\ Value)^2}$$

$$Vertical\ Pixel\ Shift = \frac{Vertical\ Reference\ Value}{c} * t$$

We entered these calculations in a spreadsheet. The following Illustration 10-13 shows the variables with a white background and all of the calculations done to get the *Vertical Pixel Shift* value for 11 x 11 examples of the 800 x 600 resolution projectors of the present setup in the Experience Cylinder.



360° Cylindrical Display			Resolution & Size (Surface)			Projector Position		Additional Values			Example
Projectors		Radius	Width	Height	Factor	$Distance_t$	Height	Sagitta	$Distance_1$	$Distance_2$	Shift at (200, 600)
6	Pixels	800,00	800	600	1,33	1466,13	780,00	107,18	1358,95	666,13	43
	mm	3000	3000,00	2250,00	1,33	5498	2925	401,92	5096,08	2498,00	160
	Factor	3,75	3,75	3,75	1,00	3,75	3,75	3,75	3,75	3,75	3,75

Vertical Pixel Shift Examples

	0	80	160	240	320	400	480	560	640	720	800
0	0	5	8	11	13	13	13	11	8	5	0
60	0	6	11	15	17	18	17	15	11	6	0
120	0	8	14	18	21	22	21	18	14	8	0
180	0	9	17	22	25	26	25	22	17	9	0
240	0	11	20	26	29	31	29	26	20	11	0
300	0	12	22	29	34	35	34	29	22	12	0
360	0	14	25	33	38	39	38	33	25	14	0
420	0	16	28	37	42	44	42	37	28	16	0
480	0	17	31	40	46	48	46	40	31	17	0
540	0	19	33	44	50	53	50	44	33	19	0
600	0	20	36	48	55	57	55	48	36	20	0

Illustration 10-13: An example of calculations made with the formerly introduced trigonometry. The six cells with a white background are the values that the calculator needs. All of the values below the headline "Vertical Pixel Shift Examples" are representations of how many pixels the pixel in question has to be shifted upwards. A graphical result is shown at the bottom.

10.1.9 Summary of Solutions

In this section we showed how to utilise the features of Unity3D to realize the possibility of presenting a 3D virtual world inside the Experience Cylinder. Several additional issues appeared along the way that were either dealt with or will eventually need to be dealt with, to create an even better experience in the cylinder. These issues primarily had to do with the construction of the Experience Cylinder. These in return triggered multiple types of distortions to the individual displays from the projectors and consequently also the whole panoramic composition. We found two solutions for the vertical pixel displacement: one approximated but implemented as a shader and another merely mathematical but potentially more precise if implemented as a shader. For the horizontal pixel displacement and as an addition to the vertical as well the simplest solution is to place the projectors in a better position.

10.2 Control

In this section we turn our focus to the second technological challenge: to get rudimentary motion control working with the Kinect mounted in a top-down position. We will start this section with background information and issues we had developing towards motion control.

10.2.1 The Microsoft Kinect

The depth sensor installed in the Experience Cylinder is a Microsoft Kinect, originally released by Microsoft for use with their Xbox 360 console. The Kinect itself is based on a reference design by another company called PrimeSense (PrimeSense 2012).

The Kinect has two cameras, a standard colour camera and an infrared depth sensing camera. Both cameras have a maximum resolution of 640x480 pixels at 30 frames per second. The depth camera works because of the built in infrared projector of the Kinect, which projects a map of infrared dots onto nearby surfaces. The depth camera can sense these dots to detect distance to these surfaces. The depth camera does have some limitations though, as black surfaces can be very hard to detect.

From its inception, the Kinect and its associated algorithms and drivers were developed with body tracking in mind, and rely on a randomised decision forest classifier in order to recognize body elements (Shotton, et al. 2011). The Kinect was trained with millions of test images in order to get its current methods and algorithms for recognizing the human body. This technique has proven very efficient [Kinect], but seems hard to transfer to our case, since the training of the Kinect is based on people facing the Kinect, and the Kinect being mounted almost directly in front of people, facing them. To get the same kind of tracking efficiency in the Experience Cylinder, we would have to train the Kinect in a similar fashion for our specific top-mounted setup and purpose, which is far beyond the scope of this project, and would likely, warrant a project on its own.

The OpenNI framework on which the tracking software was implemented includes Skeleton tracking modules and algorithms, but these are based on the same premises as the training of the Kinect was originally based on. As such, these modules were not an option either.

10.2.2 Kinect Driver Model

As the Kinect was not originally intended to be used with anything other than an Xbox 360, Microsoft did not supply official PC drivers on its release. The first drivers to enable use of the Kinect on a PC were unofficial hacked drivers, which private enthusiasts developed. Only much later, when Microsoft realized the popularity and potential of the Kinect for purposes other than console gaming, the official drivers and development kit was released.

All this puts us in a position to choose between a wealth of different drivers, frameworks and development solutions. As we had already decided on using Unity3D to develop our game prototype, we looked for existing solutions that would let us connect the Kinect with Unity3D. The most widely used solution for this was the framework OpenNI, which included a package to connect OpenNI to Unity3D.

10.2.3 Background Summary

We needed ability to control our game prototype, which could be based on the Kinect. But we had the Kinect mounted in the Experience Cylinder in a top down position. The fact that that the software for the Kinect is constructed for a front mounted position made us realise that a direct use of the conventional software was not possible. Using the Kinect would be on a premise of working with the raw data that the depth camera of the Kinect provides.

The following sections describes the solutions we found based on the aforementioned premise, in order to solve the previously discussed elements of this challenge.

10.2.4 Tracking

The development progression has been to implement recognition of certain points in the cylinder and track them with satisfying response so it is usable for gaming. The rationale for control by motion using only the depth feed is to track the players position by searching for the closest point it can detect. Barring noise errors from the Kinect, the closest point to the Kinect will be the top of the tallest element in the cylinder. In our case, this is the player.

This required us to work directly with the depth feed, looping through each frame of the depth feed to search for the highest point in the frame. Conceptually, this is simple with the Kinect mounted in a top-down position, as the point closest to the Kinect will be the highest point inside the cylinder. The highest point inside the Experience Cylinder will most likely be the top of a person's head.

Each frame of the depth feed consisted of 307,200 integers, one for each pixel at a resolution of 640x480. Looping through this amount of data at 30 frames per second was not an issue for the computer, and our code gave us a continuous output of values for the current highest point. We noticed, however, that this value could fluctuate drastically, both in terms of the actual height value as well as the position of the highest point on the grid of pixels. These fluctuations could be explained by the inherently noisy depth readings. This is a hardware related issue, so it was not possible to directly get a more reliable depth feed.

On the basis of the highest point within the Experience cylinder a virtual line from that point through the absolute centre of the Experience Cylinder could correspond to a location on the display and then be used to aim by. If the absolute centre is considered a pivot point, also represented in the virtual environment, and lifted slightly from floor height, aiming would be possible both vertically and horizontally.

Our first attempts at creating a method for the player to aim resulted in a very jittery aim that responded rapidly to all perceived changes, including those caused by noise. This form of aim was unreliable and unusable for a game where aiming was required, and we had to reconsider the way we handled the depth feed data.

10.2.5 Smoothing the Movement of Aiming Sight

Because the depth data was not directly usable in its raw form because of noise, we had to devise some way to smooth out the readings to get a more stable method of aiming. Several ideas on how to accomplish this were considered.

One idea was to make a script that smoothed out the entire depth feed before passing data along to our aiming methods. We found evidence that this had been attempted already, and even found code examples of how to accomplish such a smoothing in real time (Sanford 2012). However, when we attempted to utilise the algorithms from the code, we ran into a problem. The code was written for C# 4.0, which had introduced a number of functions that could ease the use of parallel processing. It utilised a parallelized version of a standard for-loop, which allows parallel execution of the code elements for each part of the for-loop. This is especially relevant in this case, as it allows faster execution of nested for-loops. The problem in our case was that Unity3D only supported C# 3.5, which does not include these parallelized functions. We attempted to recreate the algorithm using standard for-loops, which resulted in a catastrophic slowdown of our program that made it unplayable.

Since a smoothing of the entire depth feed was not viable within a reasonable time period, we had to turn our attention to another solution. We decided on utilizing a form of running average, so the values given to our movement script was an average of a fixed number of readings of the highest point. The number of readings we averaged over would influence the response time of the movement script, where a low amount of readings would give a fast response time but more jittery movement, while a high amount of readings would slow the response time but create much smoother movement.

This would counteract the problem of highest point readings fluctuating wildly in the height reading, but not influence the fact that the position of the highest point could also change drastically from frame to frame. But even this averaging was not enough, so we had to introduce a smoothing effect on the actual movement of the relevant game objects in Unity3D itself. Our script rotated the cannons towards a point in space, based on the height and position of the current average of highest points. It was the rotation of these cannons that was also smoothed. Unity3D included various functions to rotate objects towards a point in space, and these functions could also include a smoothing factor, that would introduce a delay on the animation of the rotation itself. The smoothing factors did allow the cannons to move fast in one direction, but sudden changes in rotation or aiming direction would have a slight ramp-up time, making most movements very smooth to the naked eye.

All this work with smoothing and moving averages did have an effect on the controls on the game, as these are a lot less responsive now, but a lot more predictable. This is a trade-off, that one has to assess for individual purposes, and in our case we valued predictability higher than response time.

10.2.6 Summary of Solutions

We found that using the raw depth feed from the Kinect was an acceptable way of realising control for games within the Experience Cylinder.

10.3 Directional Sound

In order to enhance the feeling of a virtual space beyond the experience cylinder display surface we wished to utilise the surround sound system. Game objects that come towards a player from different angles should also exhibit directional sound as to give the player an additional indication of events happening outside the player’s field of view. This section contains information about the challenges involved in using sounds in our game Meteor Defence. We will start the section with additional background knowledge pertaining to the concrete sound setup and utilising sound in Unity3D.

10.3.1 Sound Setup

At first we decided to test the ability of the surround sound setup by playing a movie file containing 5.1 discrete channels. Through this test we concluded that possibly up to 7.1 discrete channels of a surround sound movie will play in the following match between the channel row of order and channel position.

Channel Row of Order	Channel Position
1	Front Left
2	Front Right
3	Center
4	Sub
5	Middle Left
6	Middle Right
7	Back Left
8	Back Right

These associations, between channel row of order and channel positions, were not changeable in any part of neither drivers Windows nor other software in the present setup. We found that nor did the row of order match the output channel numbers chosen as connection for the speakers, neither did it match the position of speakers for a movie surround sound setup. If the speakers is named in coherence with the display number they are positioned behind, where the first display is the one where the windows start menu is and the last where the clock is, then the following list shows the connection to the M-audio ProFire 610 output channels. A coherent system was not apparent to us. That connection system proved to make everything just a bit more complicated.

Speaker/Display	M-audio ProFire 610 output channel
1	8
2	1
3	5
4	2
5	6
6	7
Sub	3

10.3.2 Sounds in Unity3D

Unity3D has a built-in system to handle directional sound. Sounds can be attached to a game object which is able to move and be positioned anywhere. A sound listener, typically attached to the camera, can also move around. The distance between sound and sound listener can be calculated so sound volume can be adjusted accordingly. Also the direction can be calculated and used for mapping the sound to the correct speaker up to the quality and extent of a 7.1 DTS surround sound system.

In the Experience Cylinder setup we had no success in using the sound system of Unity3D. It seemed that Unity3D and the sound card drivers had compatibility issues. Other tests performed on another but fully capable 6.1 DTS setup also proved unsuccessful. This setup consisted of a Windows based PC with an onboard 8 channel AC'97 HD sound card attached to a 7.1 DTS HD amplifier with speakers. Due to internal bandwidth limitations in this computer it delivers only up to 6.1 DTS even though all other hardware is capable of full 7.1 DTS HD. This was tested with 5.1, 6.1, and 7.1 movie clips. But still all sound tests performed in Unity3D resulted in stereo only, but directional stereo at the least. Thus we determined that using the built-in sound system was not possible at this time, and another solution had to be found.

10.3.3 Background Summary

In order to get directional sound and make it easier to make changes in all software involved in sound, we had to find other solutions. This involved changes to the sound setup and finding alternatives to the Unity3D sound system. The next sections are about the solutions we chose and how we solved the issues at hand.

10.3.4 Sound Setup

Although the following solution was a minor alteration in the setup, it simplified consecutive modifications and adjustments to the software involved. The steering group approved of an alteration to the setup, so both input and output channels were in the same order and thereby numbered label. Then every channel number through the system matches, both by internal channel numbering, order, and by different hardware labelling. Though in a 6.1 setup the two last channels of a 7.1 source are typically merged into one channel. How the whole setup was matched to the six displays, was a relatively minor decision. We decided that the center speaker should be placed between display 3 and 4 but since the speakers are positioned in the centre of displays and not in between them, we chose to just use speaker 3 behind display 3 as the center speaker. It could have been any of them.

Channel Row of Order	Channel Position	Speaker	Display	M-audio ProFire 610 output channel
1	Front Left	1	2	1
2	Front Right	2	4	2
3	Center	3	3	3
4	Sub	Sub	n/a	4
5	Middle Left	5	1	5
6	Middle Right	6	5	6
7	Back Left	7	6	7
8	Back Right	7	6	8

As an example the following Illustration 10-14 shows the setting within the *m-audio driver interface*. The pane of output channel 1 and 2 is selected. Here one should select which internal software return-channels should pass through to output 1 and 2. Because of the new setup the selection of software return corresponds in number value. Software return 1 is directed to output channel 1 and likewise by 2 and every other channel in the additional panes according to the list above.

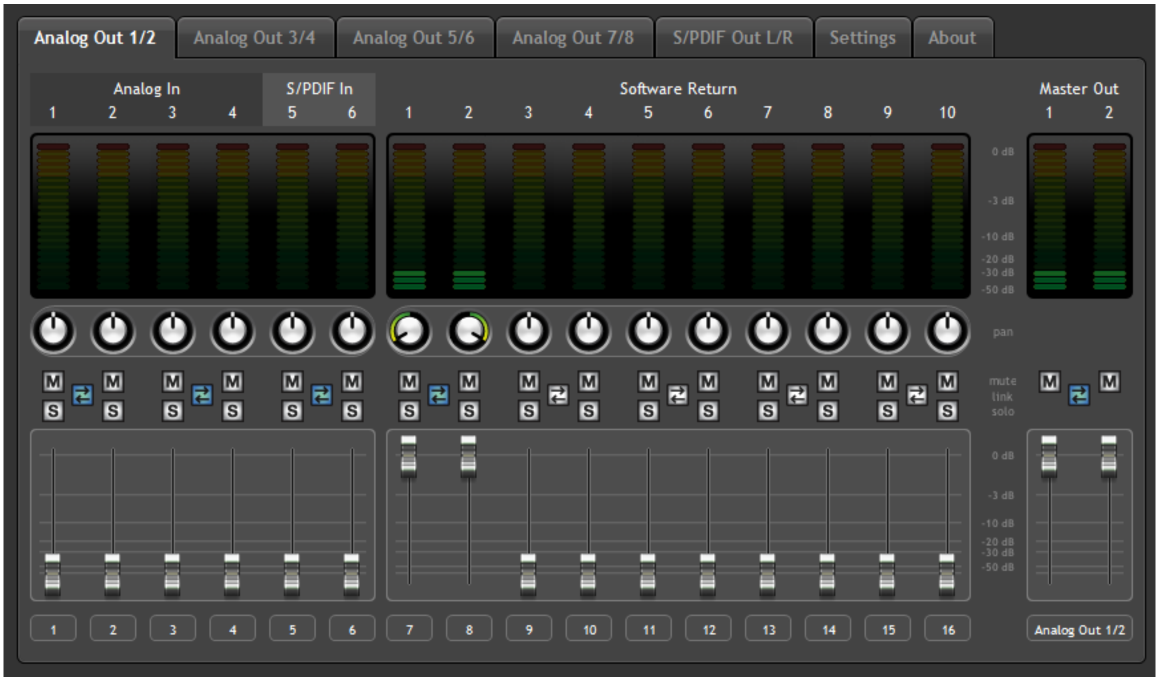


Illustration 10-14: Screenshot of the driver software for the M-Audio 610 ProFire Soundcard. The pane of analog output channel 1 and 2 is selected and here input channel 1 and 2 is selected as the actual output.

10.3.5 Directional Sound

Since we had trouble getting real surround sound from Unity3D to the sound system we had to come up with another solution. The technical staff used *Max 5* for their directional sound setup in Mac OSX. But after their decision to collaborate on using Windows we jointly decided to use the same output setup. Their usage of *Max* was specific to *The Sea Stallion* and at this time they were focused on porting the main functionality to Windows. They were still helpful enough to advise us about a third party function for Max that was able to calculate sound distribution to individual speakers in a 3D sound setup (Pulkki 2000). This third party object was created by Ville Pulkki. We had to figure out how to use that in Max and how to connect Max and Unity3D.

10.3.6 VBAP

The following Illustration 10-15 shows the theory behind Vector Based Amplitude Panning (VBAP) created by Ville Pulkki. He utilises what he calls the triplet-wise panning paradigm which states that a sound source panned equally out to three speakers places the sound source virtually in the listening direction at the centre between them.

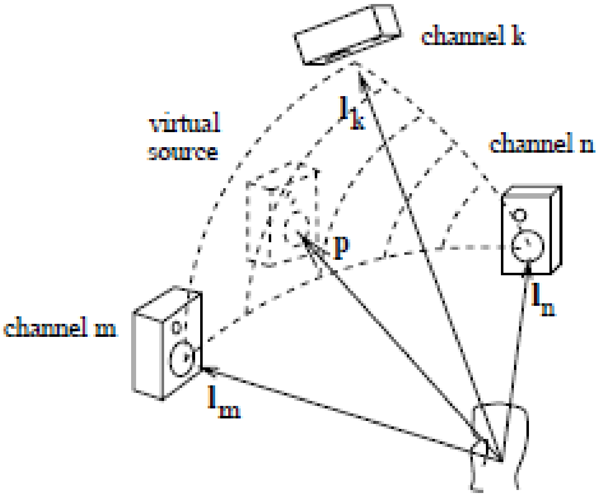


Illustration 10-15: The triplet-wise panning paradigm states that a sound source panned equally out to three speakers places the sound source virtually in the listening direction at the centre between them.

With this theory it is possible to place and define positions of multiple speakers to create a network of possible directions a sound can have. An example of this is shown in the following Illustration 10-16 where multiple sounds is passed through the VBAP and played to the listeners from the calculated directional origin even though no speaker is positioned exactly there.

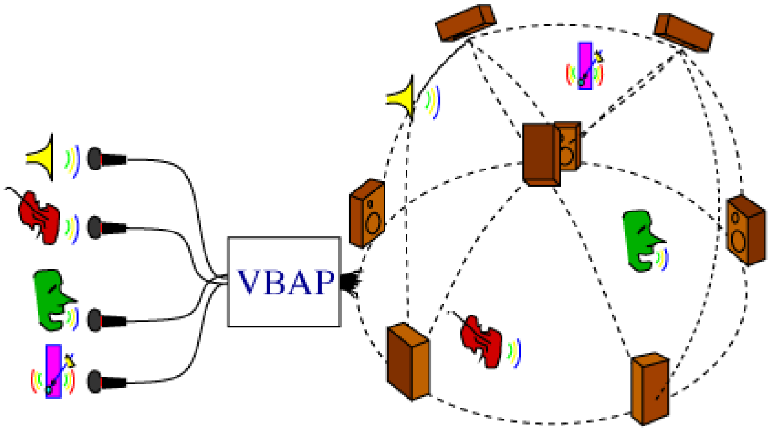
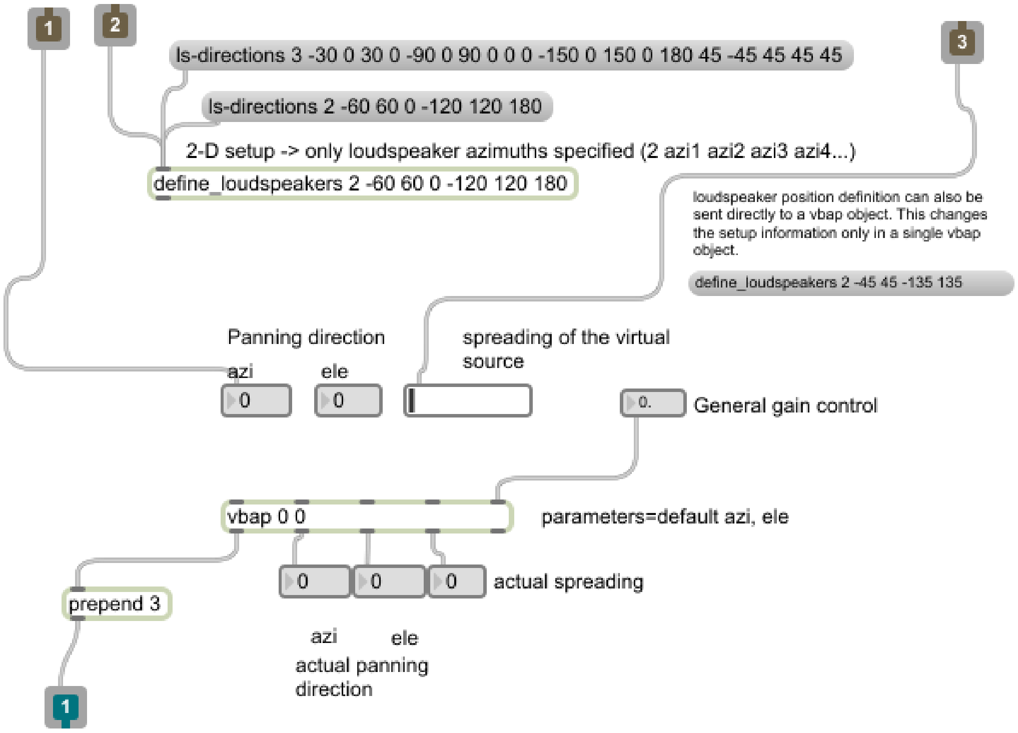


Illustration 10-16: Multiple sounds is passed through the VBAP and played to the listeners from the calculated directional origin even though no speaker is positioned exactly there. (TKK Acoustics Laboratory 2006)

In Max the original patch looks similar to the following Illustration 10-17. We defined it as a subpatch for our Max 6 solution and added the three inputs shown at the top, and the one output at the bottom. Inputs 1 and 3 receive data about azimuth and spreading respectively and input 2 makes it possible to activate the speaker definition from outside the subpatch. The VBAP object needs the speaker definition set every time the program is started. Being able to do it without entering the subpatch makes it more accessible. The number sequences that starts with 2 is ours as well and defines that 2 dimensional speaker definition which is the direction/position of our six speakers. Additionally the prepend object were added to make the sound source in question target a specific output. The prepend object simply adds the number defined to the beginning of the output. In this case it is 3. In other cases it could be something else. It is used to make following objects able to distinguish between origins. The specific use is explained in detail later.



ls-directions 3 -30 0 30 0 -90 0 90 0 0 0 -150 0 150 0 180 45 -45 45 45 45

ls-directions 2 -60 60 0 -120 120 180

2-D setup -> only loudspeaker azimuths specified (2 azi1 azi2 azi3 azi4...)

define_loudspeakers 2 -60 60 0 -120 120 180

loudspeaker position definition can also be sent directly to a vbap object. This changes the setup information only in a single vbap object.

define_loudspeakers 2 -45 45 -135 135

vbap 0 0

parameters=default azi, ele

prepend 3

Illustration 10-17: This is the VBAP object created for and shown in a Max patch. This patch has three inputs and one output. Input 1 is the value of the sounds position by azimuth. Input 2 makes it possible to activate the define loudspeaker object from the parent patch. Input 3 is the value needed for adjusting the spreading of the sound, meaning how wide a direction the sound would seem to origin from. In the end the output delivers the directional data with the value of three in front. This is done by the prepend object.

10.3.7 µ Max-Unity3D Interoperability Toolkit

In order to get data from Unity3D into the VBAP patch in Max we needed some kind of connection between them. We found *µ* to be the solution we needed (Bukvic and Kim 2009). On the Unity3D side, *µ* is a set of files that makes it possible to define: the objects in Unity3D that transfer data and the data they transmit, along with files containing the code that performs the actual transfers. On the Max side it receives the data and routes it to the correct place according to names of game objects and data type from Unity3D. This is visualized in Max as in Illustration 10-19 of our main patch. The main patch contains more than what comes from *µ*. In the main patch *µ* amount to the content from the *netreceive* object at the top to green objects as seen in the illustration. There are two routes from the *netreceive* object because we have two sound sources.

10.3.8 netsend & -receive

On the Max side, *µ* makes use of two third party objects called *netsend* and *netreceive* (Matthes 2005). These are the objects that make it possible for Max to either send data to or receive data from other software solutions over the TCP or UDP network protocols. We needed only the *netreceive* object though in order to read data from Unity3D in Max. The following Illustration 10-19 shows the *netreceive* object at the top.

10.3.9 Main patch and Additional Sub Patches

The two sounds we decided to implement in our game were: a sound when a meteor explodes and the sound of the two machine cannons. The explosion had to sound louder the closer to the centre. If it reached the actual centre the spreading should be maximized, which means that all speakers in all directions would play the explosion. This way it could possibly feel more like you were hit by it. The objects in Illustration 10-19 after the three green value objects to the left are calculations that translate the coordinate of the explosion to a direction and a distance as well as functions that translate the distance data to the values needed for volume and spreading. To the left where the data for the cannons orientation comes in, there is only a need for adjusting the rotation. In the top is an object called *loadbang*. It has the ability to activate all its connections when the patch is loaded. This means that it can start all the settings we need done right away. This includes loading sound files, adjusting volume and spreading, and applying the definition of the speakers inside the VBAP objects.

The two objects in the top called *patcher*, which are sub patches, contains the loading of the sound files. The following Illustration 10-18 shows the patches. Notice how the explosion is not looped, but the machine cannon sound is. This is marked by loop 0 or 1 for no looping and looping respectively. These patches were based on an internal Max 6 example.

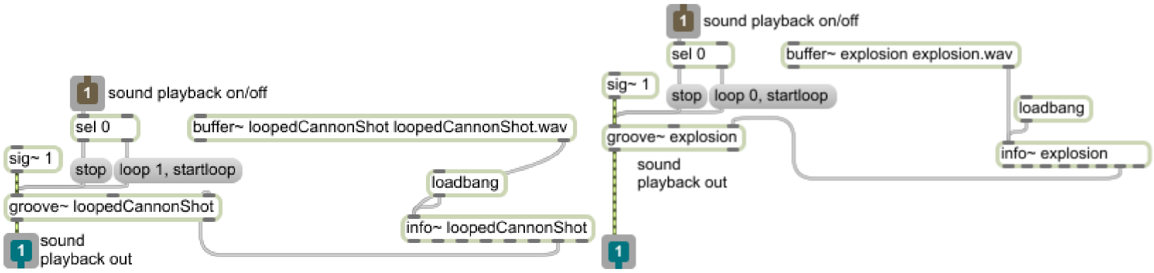


Illustration 10-18: Two different patches that loads the sound files for use in Meteor Defence. The one that loads the loopedCannonShot.wav file has a loop value of 1 and will therefore play in a loop. The explosion patch has a loop value of zero.

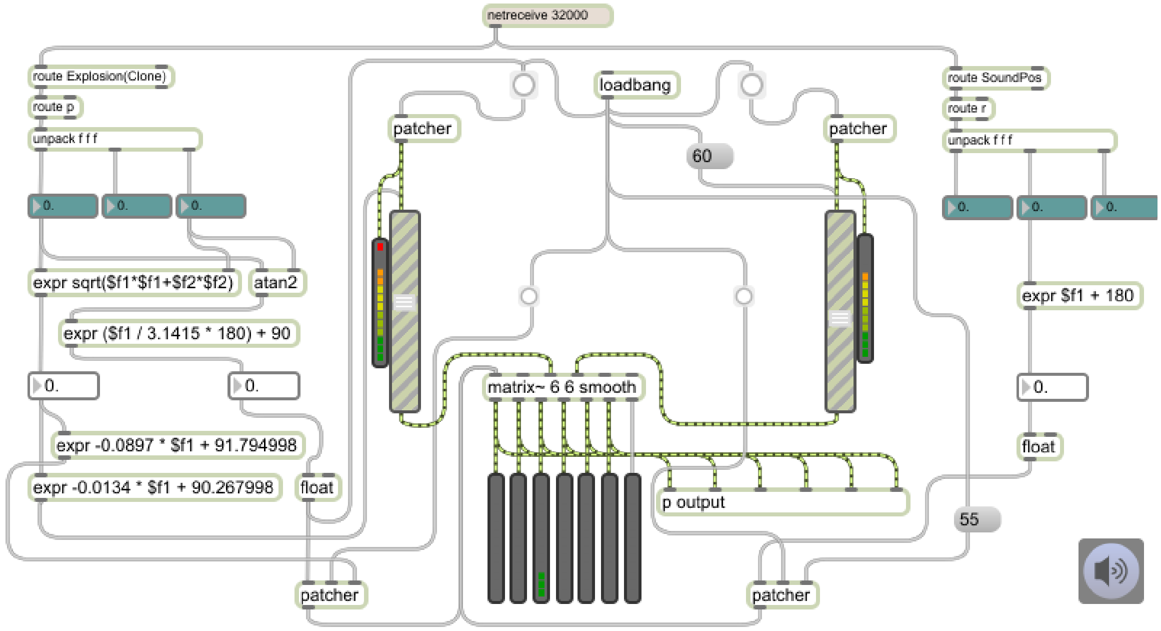


Illustration 10-19: This is the main patch of our solution. To the right and left are two similar setups. These are the Max part of *μ*. They receive data from *the μ part of* Unity3D. Two objects in Unity3D send position data. This data is send to the VBAP patches in order to calculate the position. That position is transferred to the matrix~ object, which mixes the sounds together.

The *matrix~* object mixes multiple sounds and this is where the previously mentioned *prepend* object of the VBAP sub patches comes in. The two bottom *patcher* objects contain each of the VBAPs. They send their data to the same input but with a different *prepend*. The sounds also enter different channels on the matrix. But the channel number matches the *prepend* number. So the explosion sound enters channel 2 and the VBAP of the explosion has a *prepend* value of 2 and this is how the matrix knows what to mix. The solution to use two VBAP objects was quick and dirty because according to the VBAP theory it should be possible to control multiple instances with only one.

With the aid of Nikolaj Møbius from the technical staff, a sub patch was additionally created, which contained a crossover filter in order to only send low frequencies to the subwoofers and higher frequencies to the satellites.

10.3.10 Editing in Audacity

As formerly mentioned we found the sound files on the royalty free sound website Soundbible. The explosion sound required no alterations, but the cannon sound did. First we did not find a loopable sound, and secondly our cannons in the game ended up as being a mix of visually a long range, high caliber cannon and functionally a small, fast machine gun. So all the machine gun sounds were a little too light and we tried to make it sound a little heavier although still machine shooting. To create the looping we cut down the sound to a point where it sounded somewhat even from start to finish and mixed the end of it on top of the start to create a more seamless loop.

10.3.11 Summary of Solutions

Although a bit more cumbersome than just using the internal sound system in Unity3D, we managed to find an alternative solution to directional sound. This solution involved software from three different developers and a lot of work trying to figure out how to use it.

10.4 Creating Game Elements

In order for a game to be realized it needs to contain game elements. Firstly we would like to present the background information we needed in order to create the different objects and elements we added to our game prototypes.

10.4.1 Physics

As we progressed with learning the basic functionality in Unity3D, we started to examine, how to create movement of objects in the virtual environment. There seemed to be two different ways to handle movement in the game: either by animation through the use of scripting, or with the included physics system augmented by scripting. We began working with both possibilities in parallel, and while pure scripting was very easy to get started on, it was not easy to get the expected results. Additionally, this method did not include a way to handle gravity or even collision between objects, so it was quickly abandoned. The physics system had both of these possibilities, but it was frustrating to get started on, as different tests did not go exactly as planned. It was easy to generate some movement, objects were affected by gravity and collision was handled very smoothly. But the accuracy of movement was wrong, and as we desired the possibility for accurate movement, we had to learn more about the physics system and refresh our knowledge on classical mechanics.

10.4.2 The Unity3D Physics System

The physics system in Unity3D is based on the NVIDIA PhysX engine (Nvidia 2012). So movement and collision in Unity3D can be handled in a way similar to the way movement and collisions work in the real world: applying force to objects. The force can be continuous, like gravity or the jet engine on a rocket, or instantaneous, like firing a bullet from a gun, or kicking a ball. The same concepts can be applied in Unity3D. In Unity3D you create a virtual environment that can use these basic concepts, so movement can be created by applying force to objects in the virtual environment. Not all objects in the virtual environment have to be affected by the physics system. It is possible to create objects that ignore gravity or collisions with other objects.

After some initial attempts at scripted animation in Unity3D, we decided that it would be simpler to handle movement of objects in the game with the physics system whenever possible. Since our initial game idea was based on some pretty simple concepts, this was conceptually straight-forward to do.

The physics system in Unity3D was a problem area in our initial work to understand the Unity3D engine. We started out by getting re-acquainted with classical mechanics and physics. We then started working in Unity3D, constructing simple examples and comparing observed behavior with intended behavior. Almost immediately we ran into problems, as some functions and constants in Unity3D did not have a listed measuring unit. A simple example with objects being launched specific distances using only a vector (to indicate direction) and force, did not always give the expected results. The official Unity3D documentation did not offer any explanation, but through Unity3D message boards we learned that the length of the vector influenced the length of the launched objects' ballistic path. To get precise results from our calculations, the vector indicating launch direction had to be exactly length 1.

Unity3D has a built in method to normalize any given vector, which reduces the total length of the vector to 1, while preserving the direction of the original vector. We attempted to use this normalize method, but ran into inconsistencies and rounding errors, and decided to calculate the relevant unit vectors directly based on their angles.

10.4.3 Scoreboard and Lives

Since we did not manage to make our game ready for multiple players we had to create other means of facilitating gameplay and quantifiable outcome - to make the prototype an actual game. This was realised by creating a system for registering, keeping, and showing game progress.

In most games the player looks in one direction, the direction of the display. In our case when the display is all around you it does not make sense to add the typical layer on top to show the scoreboard. There is no predefined viewing direction of the player and thereby no specific position for the scoreboard to be placed so another solution had to be found.

10.4.4 Background Summary

In order to create our meteors we needed to work out how to use the Unity3D physics system to create realistic movement of said meteors. Additionally, we presented the problem of displaying the state of the game, within the player's field of view, on a 360° screen

Following are the solutions we developed in order to create our game elements.

10.4.5 Trajectory Calculations

The starting point of our calculations is seen in Illustration 10-20. Using the relationships between the sides and our know variables, we can calculate x, y and z, which are needed to denote the unit vector in Unity3D.

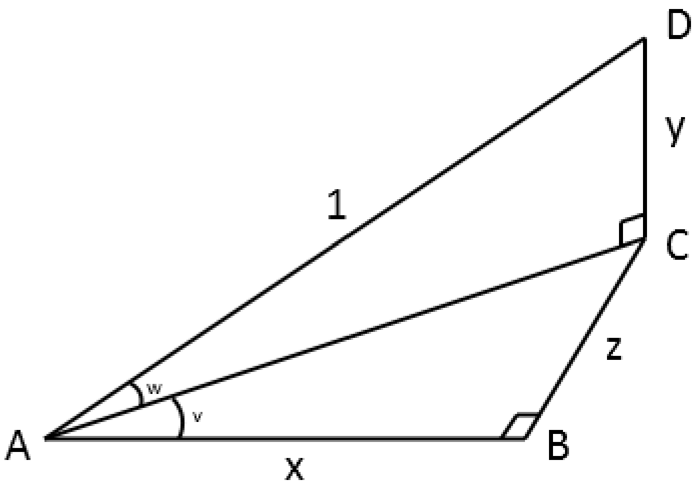


Illustration 10-20: Two right triangles indicating the projections of the unit vector |AD| onto the x, y and z-axes. v is the angle along the horizontal plane, and w is the launch angle.

In our case, the angles, v and w, are known along with the length of |AD|, which is 1. From this we need to calculate the lengths of x, y and z. From the Pythagorean trigonometric identity, given a right triangle, we have:

$$\sin\theta = \frac{opposite}{hypotenuse}$$

$$\cos\theta = \frac{adjacent}{hypotenuse}$$

For the ACD triangle, we know the length of the hypotenuse, 1, and θ, which is w. The length of y can thus be determined by isolating the opposite:

$$y = \sin w$$

The length of |AC| can be determined by isolating the adjacent:

$$|AC| = \cos w$$

The length of |AC| can then be used to determine x and z, by using the length of |AC| in the ABC triangle. The same trigonometric identities can again be used. To determine z:

$$\sin v = \frac{z}{|AC|} \Rightarrow z = \cos w * \sin v$$

$$\cos v = \frac{x}{|AC|} \Rightarrow x = \cos w * \cos v$$

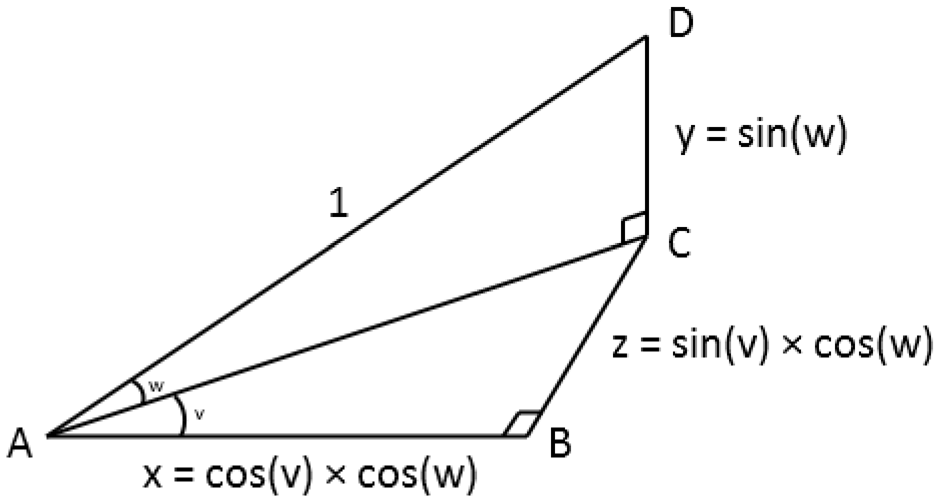


Illustration 10-21: This illustration shows the formulas needed to calculate the x, y and z-coordinates to to create a unit vector with the direction indicated by the angles v and w.

To launch an object in a ballistic trajectory towards a specific point, we also needed to calculate how fast the object needed to go in order to land in the correct spot. In Unity3D, this can done either by applying impulse force to an object, meaning a single instantaneous application of force, or by setting an instantaneous velocity change on the object. If force is applied, the mass of the object is taken into consideration, meaning more force is required to move heavier objects, while velocity change does not take object mass into account. Air resistance could also affect the object, but we chose not to use air resistance in our game. Since we needed all objects to land in the same spot, regardless of how heavy they were, velocity change was deemed the best fit for our needs. We found a formula to calculate horizontal distance travelled for an object in a ballistic trajectory. For situations where starting point and target point is on the same level, this formula is:

$$d = \frac{v^2 \times \sin(2\theta)}{g}$$

Illustration 10-22 below depicts a meteor trajectory as created in our game by the formula above. The meteor travels from a point on the circumference on the circle and hits the target point in the centre.

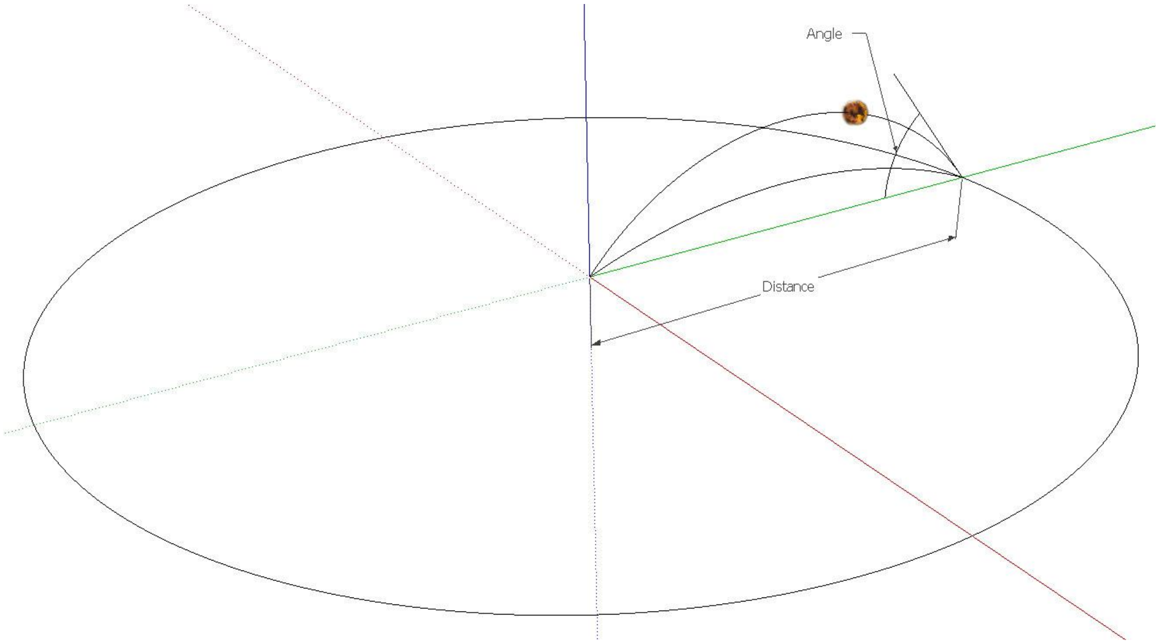


Illustration 10-22: This is a model that visualizes two possible trajectories of the meteors from one origin. Values for the vertical angle and distance is used to calculate how much power is needed to make the meteor hit exactly in the centre.

For our purpose we wish to isolate the speed; *v*, as this is the only unknown variable for our specific purpose, since we know how far we want the object to move and we decide what angle it gets launched at.

$$v^2 = \frac{d \times g}{\sin(2\theta)}$$

$$v = \sqrt[2]{\frac{d \times g}{\sin(2\theta)}}$$

These calculations were used to make objects fly towards the exact same point in the virtual environment, regardless of where the objects were created. Another algorithm was used to randomly generate the objects on a circle around a central point. We could in principle have spawned the objects anywhere on the horizontal plane where the target destination was located. The algorithm simply needs to know which horizontal and vertical angles to launch the object towards and the total horizontal distance the object needs to travel before impacting the horizontal plane it was launched from.

10.4.6 The Meteor Game Elements

The trajectory calculations just shown were used to send the meteor elements on their correct path, but the meteors still seemed very dull.

The meteors were a simple gray colour, as no attention had been paid to their visual appearance. We decided to add a rock texture to the meteors to make them a bit more realistic. Additionally, we decided to add particle effects to the meteors, to make them appear to be engulfed in flames. This helped create the illusion that they were moving at fast speeds, and helped highlight the meteors to the players. After these purely visual alterations were performed, the meteor still seemed dull.

It was clearly visible when looking at the meteors in the game that they did not rotate around themselves in any way. It was decided, that the meteors needed to rotate around their own axes at random speeds, to make them seem more different, even though they all had the same size, textures and particle effects. Giving the meteor objects rotation helped make them appear more like physical objects than just constructs in a virtual environment.

10.4.7 The Cannon Tower

This game element is composed of several individual parts. Most of these were modelled by us in SketchUp. The following Illustration 10-23 shows the two cannons mounted to the cannon turntable. Additionally the frame for positioning the scoreboard is attached. We found it important to create all elements in one SketchUp file and then export them to individual Collada files. This way it was easier to control the positioning of the individual elements in Unity3D. Furthermore the cannons include two laser aiming sights attached in between the cannon tubes. These are included to assist the player in aiming the cannons, as hitting the meteors without the aiming sights which can be quite difficult.

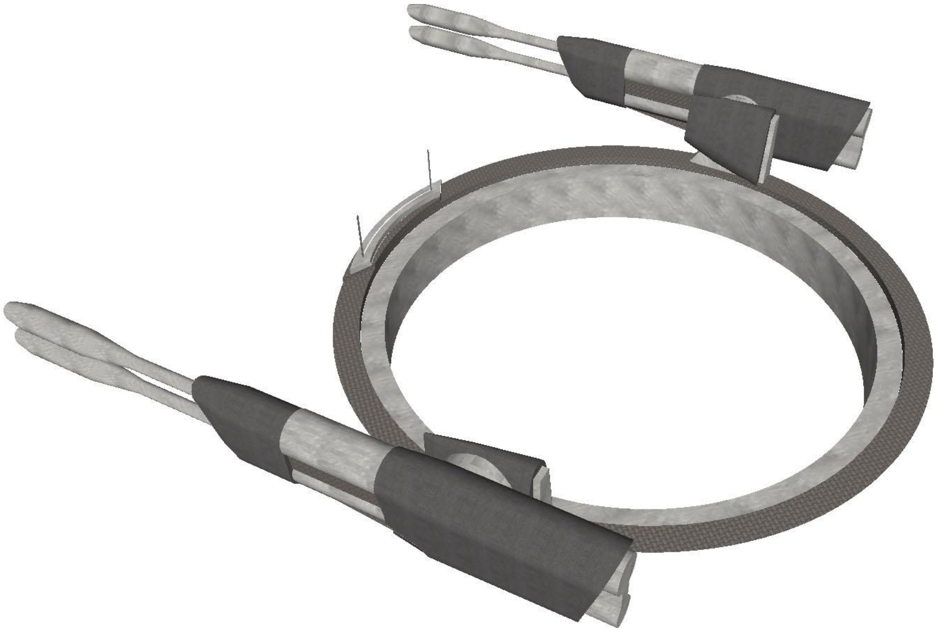


Illustration 10-23: The cannon turret for our Meteor Defence game modelled in SketchUp.

10.4.8 RUC3D

These game elements consist of all the buildings and the terrain. The buildings were created by (Thorlund, et al. 2009). SketchUp was also utilised to create these. The way SketchUp handles textures added difficulty to creating the terrain, so a temperate black and green terrain was created instead. In this project we found that Unity3D has the needed feature of being able to add the terrain with a texture and then manipulate the contour afterwards while the texture stays in place. We added a new terrain to RUC3D by using a picture from Google Maps as the texture. The image resolution is low which results in an apparent visual difference between buildings and terrain.

10.4.9 Implementing Scoreboard and Lives

The idea we came up with was to add a scoreboard physically to the construction of the cannon tower. Since the cannons rotated around, why not in addition to that add it to the rotational parts in order for the scoreboard to always be in the present direction of view, see Illustration 10-23. This however spawned the additional task to actually implement an output of internal variables values in the virtual environment while being able to move them around. We fixed this by utilizing the feature called camera to texture. This feature makes it possible to render any part of the scene to a texture/surface on a polygon inside the scene. In an area not visible to the 360° camera, 3D texts displaying the user data were placed. A camera set to render to texture was put in place to film the 3D texts. The filmed data was then sent to the texture and applied to the scoreboard polygon mounted to the cannon turntable.

11 Description of the Finished Prototypes

The primary game prototype that has been the vehicle for development we call Meteor Defence. It can be compared to a stationary 3D version of the classic arcade game *Missile Command* where the players have to defend a base from incoming missiles by shooting them down.

As a secondary prototype we have implemented a 3D model of RUC in which a player can experience a stroll on RUC campus. We have named the prototype *A Stroll on RUC.*

11.1 Meteor Defence

The game prototype Meteor Defence has its setting in outer space. This is seen in Illustration 11-1 below by the small dots in the background depicting a starry sky. The orange objects are the approaching meteors. They emerge in different places on the horizon and travels towards the cannon tower. The game objective for the player is to shoot the meteors before they impact the cannons. For every destroyed meteor the player gains a point and for every missed meteor the player loses a life. The player's current total points gained and remaining lives are displayed on the scoreboard. The cannons are controlled by the player's movement around the centre of the cylinder and aiming upwards is done by ducking. The cannons fire continuously, so the player is only responsible for aiming. When meteors are destroyed an explosive sound is played. This sound is louder the closer the meteors are to the cannon tower when they explode. Additionally the explosion sound comes from the same direction as the meteor.

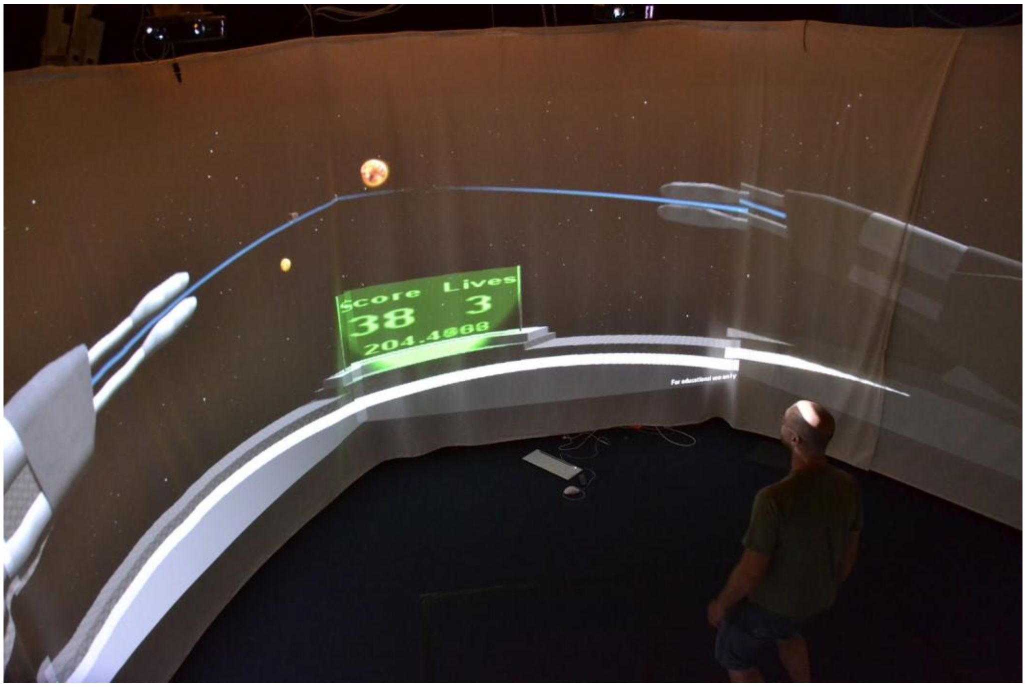


Illustration 11-1: A picture from inside the Experience Cylinder while a person is playing Meteor Defence.

11.2 A Stroll on RUC prototype

As a secondary prototype we have implemented a 3D model of RUC campus in which the player can walk around and look at the different buildings from the outside, see Illustration 11-2. It is not an actual game, but a supplement to the first prototype. With the *A Stroll on RUC* prototype we can experience a virtual environment with a conventional 3D world model. The player can walk around the campus by moving away from the centre of the cylinder and otherwise stand still in the centre. A step to a given direction results in continuous movement in that direction. The experience of moving with this kind of control can be compared to standing on a flying carpet as virtual as that might be or a Segway.

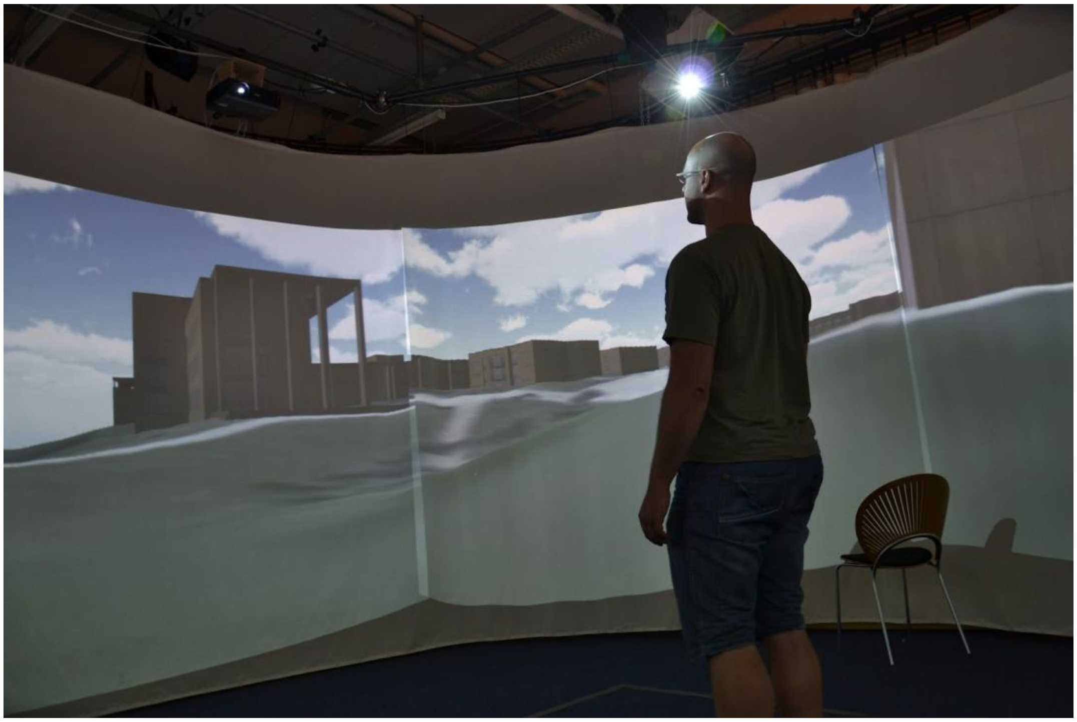


Illustration 11-2: A picture from inside the Experience Cylinder while a person is trying a simulated walk in a 3D model of Roskilde University campus (A Stroll on RUC).

12 Analysing the prototypes through our model

In this section we analyse our prototypes in the light of our model of immersion. This is done by describing the prototypes with respect to each concept in the model. To explain the different connections between a game session and a player's reaction, we will also include a few informal observations we have done. During the development we have had a small amount of people of varying ages trying out the two prototypes.

We begin by looking at the system and the relation between form and content. In our case the Meteor Defence prototype is carefully designed with the design of the Experience Cylinder in mind. Still we differentiate between *form* as the physical platform combined with the underlying software platform that makes it possible to run and play the game prototype. *Content* accounts for the application type which in our case is a game as well as the specific design and configuration of the game type which has resulted in the Meteor Defence game prototype.

Regarding *immersion* we can consider the different technologies and how we utilised them in order to describe how we have utilised the Experience Cylinders technical capabilities to support immersion.

The first thing to notice is how well the game prototype makes use of the display. The player is attacked from all angles and has to keep focus at all times in order to keep track of which meteor to hit next. This requires for the player to look at the display at all times. The sound system and the way we utilise it to support the actions in the game is also a means for attracting attention thereby leading to a degree of presence.

By utilizing the Kinect, players engage in a way they have not tried before. It is physical, so they have to be aware of their whole body movement, and they have to be so continually, which is again a way of keeping attention towards a central aspect of the game potentially leading to involvement. On the downside it can also be too much to the player, and physical discomfort from the new type of game control may lead to dissatisfaction and break the involvement.

The inherent structural parts of the game prototype mainly concern the spatial structures and the gameplay of the game in our case. In the Meteor Defence prototype we have strived to present nice looking graphics for the most important part; the meteors. As the game world is placed in space we have no traditional 3D environment as with the *A stroll on RUC* prototype. This means that the player can put more focus into the gameplay as well as the physical interaction. One thing is to learn how to control the game and another is to actually deal with the game objective, which is to shoot as many meteors as possible before all game lives are lost. This design mainly supports involvement concerned with the gameplay.

People not playing, but watching the game sitting on the floor in the periphery of the physical space, tend to advise the playing person where to look for meteors. We can also see that a single play session can be considered part of a bigger social context when more players take turns playing. As with normal single player games it is possible to have spectators and eventually take turns. From this a communal game arises as the players want to beat each other's high scores. Players in competition add meaning to the game. It is not just about shooting meteors and gaining points. We suggest that this could be a precursor for social presence since the competition helps adding meaning to the game.

We recognize the ability of a game session to saturate the player's sensory apparatus possibly resulting in a physical alert condition and mental attention towards the task at hand. The player's detection of meteors that move in a recognisable way and the understanding that the meteors actually moves toward the player supported by the sound effects would be a sign of environmental presence. The concept of presence understood as an effect of immersion.

We have also noted that it matters to some players to shoot the meteors and that the relation between the gameplay and this internal motivation can be a sign of involvement. Involvement understood as a combinatorial effect of inherent game structures and the player's internal motivation to engage in these structures

One thing which is missing, to claim all aspects of involvement is *emotional* attachment. We have seen that it matters to some players to shoot the meteors as soon as they begin a game session but we cannot be sure of the reason why they do so. Different players have different reasons to play determined by their personality and personal profile, and especially within our case, their experience with games in general.

Our game prototype might afford some degree of environmental and social presence, and involvement concerning gameplay, but not involvement in the emotional sense.

13 Discussion

In this section we treat the dominant areas of discussion which concerns the creation of the model, the audiovisual technological foundation for the game, and considerations about a control interface for the game. We refer to the problem statement in section 1.1 to explain the reverse order we discuss each of the statements, beginning with the model, then the challenges and finally the connection between form and content.

Concerning the model, in chapter 6, Building a Model of Immersion, we presented a model of the concept of immersion appropriate to the context of the Experience Cylinder. In the model we drew on ideas from the literature in virtual reality and games. These areas provided valuable insight to the understanding of creating immersive experiences, but seemed at odds with each other regarding the concept and definition of immersion. Unifying the two areas into a model for evaluating an immersive virtual reality game experience seemed unrealizable, until we decided to use the concepts of form and content, as a starting point.

Game literature takes little or no consideration of the form of an experience, in the sense of the way it is delivered to the user. Additionally, game literature largely ignores the fact that a game experience can change drastically on different devices with different sound setups or different monitor sizes. One could argue that game design literature should treat game development for different platforms, but it does not.

Virtual reality literature does take the form into account, but can treat the content of the experience in a rather shallow manner, as it appears to mostly deal with influencing users rather than involving them. In virtual reality literature, there seems to be an implicit expectation, that the user is interested in the content being delivered to them.

The novel aspect about our model is that it clearly distinguishes presence from involvement, which is important and not made clear within the literature on the concept of immersion within games. The model also separates different aspects of an experience, and unlike the concept of immersion within games it takes the whole system into account by making a distinction between form and content.

The limitation of the model, see Illustration 6-3, is the indication of a linear connection between:

$$form \rightarrow immersion \ \rightarrow presence$$

$$and$$

$$content \ \rightarrow game\ structures \ \rightarrow involvement$$

This is too simplistic. It would be more correct to note, that these connections are the primary ones, but an argument can be made for secondary connections from immersion to involvement and from game structures to presence. The mere quality and style of a virtual environment could stimulate a player's curiosity leading to further investigation of the virtual environment which would be considered involvement. This involvement could also reflect back on a player's sense of presence (Slater 2004). Involving content can distract the player from finding flaws in the experience, flaws that could lead to a loss of sense of presence.

So while the model is useful to get an overview and better understand the individual elements that constitute an experience, it cannot be used to identify any synergetic effects between a user's experience of presence and involvement.

One needs to utilize the model to subdivide and analyse an experience in order to determine if everything was combined in a meaningful manner and where improvements are possible. "How can the gameplay elements best utilize the platform?" and "how does one avoid being limited by the platform when delivering the content?" are two central questions to ask.

Alternate applications of the model are still something that needs to be tested. We have utilized the model analytically but utilizing it as a design tool, to create an immersive game in a virtual reality environment like the Experience Cylinder could be interesting to investigate.

The audiovisual technological foundation and particularly the challenge of displaying a 3D world on a 360 display, was presented in chapter 10, The Development Challenges, which lead to a discussion about the applicability of the specific improvements we came up with.

The work we did to understand projection correction resulted in a visual improvement to our game prototypes. However, in the end more precise solutions will offer no additional improvement if they are created under false pretences. If a solution is created for a geometrically cylindrical display, then the improvement is less significant if the display is even slightly differently shaped. In our case, the hexagonal distortion of the display shape and the folds of the canvas diminished the effect of our solution, but the improvement was still noticeable.

The general problem of projection correction highlights one of the main benefits of using a box-shaped CAVE rather than the Experience Cylinder. The flat projection surfaces in a CAVE require little to no display correction, and aligning the projectors is a one-time operation. However, had the display surface of the Experience Cylinder also been a rigid material and been of an accurate cylindrical shape, projector alignment here would also have been a one-time operation.

With a rigid display surface, the problem of folds on the display surface would also be completely removed. The folds in the display surface had a very negative impact on the way moving objects was displayed. More so than static objects, movement highlighted the folds of the canvas resulting in a wavy distortion. This effect was most noticeable in the *A Stroll on RUC* prototype where the walls of buildings were distorted as you moved past them.

A shader based on our trigonometric calculations would be applicable to a rigid and cylindrical display but the shader applied in our game prototypes is effective for both the current basic display shape and projectors. A change to short throw projectors would however require modifications to the shader.

The 360° camera setup could prove to be unique to Unity3D; we did not examine how to do the same in other game development tools. The shader on the other hand could be applicable in other software solutions. primarily because of the multi platform shader programming language CG. The use of the shader is predicated on the ability to apply the shader to an area corresponding to each projector.

In chapter 10.2, Control, we described the challenge in utilising motion control. Recall that the Kinect is mounted overhead and that the readings from it are noisy. Our implementation of motion control fulfilled its overall purpose, albeit at a suboptimal performance level. This can be largely blamed on the quality of sensor readings, but one cannot help wondering whether the sensor readings from the Kinect are so noisy, that it would be better to utilize the existing functionality of the Kinect in the cylinder, and largely avoid the problem of noisy readings.

The OpenNI package has built in body recognition and functionality to determine the positions of individual players in a room, all of which function despite the noisy sensor readings. However, this functionality is built for a forward-facing Kinect Sensor. It is conceptually harder to determine the positions of players in the cylinder with a forward-facing Kinect sensor, but this is mostly based on the premise of starting from scratch and creating this functionality from depth readings. The possibility is there to utilize the work other people have made in optimizing tracking for the Kinect, which could reduce the difficulty drastically. On the other hand, with the top-mounted Kinect sensor, the calculations to track users are very simple. But is this simplicity actually a benefit? As we found during our work with the Kinect, the problems regarding user tracking and motion control were mostly related to the quality of the depth feed.

Our cursory look at improving on the depth feed readings showed, that the calculations needed to smooth the depth feed were not possible in our development platform. By changing our development platform we might had been able to smooth the depth readings, but this would still put us behind existing functionality in OpenNI.

Ultimately, our motion control worked, but if we were to do any further development on our work, we would move the Kinect to a different position to utilize OpenNI more effectively and get more advanced motion control functionality.

We once more return to our model as presented in chapter 6, Building a Model of Immersion, to discuss the relation between form and content. This is to answer how the Experience Cylinder can be used to support immersive game experiences.

The display and sound technologies, as part of the form, are immersive technologies that can amplify the persuasive aspects related to the content. This means that the technology amplifies the quality of the content. Positively if the content is stimulating, and is delivered in a sensorially convincing way, but the opposite could also be the case. If the content is stimulating this could contribute degree of presence.

We have found that controlling a game through physical movement - with a focus on reflexes and response time - is a good way of laying the foundation for gameplay, which in turn can cause player involvement. Displaying the current state of the game and keeping it visible at all times to the player is a good way of keeping the player aware of game progression: the challenge is to implement it on a platform where the player changes looking direction often.

We did not investigate the foundation for letting the Experience Cylinder support narrative but *The Sea Stallion*-installation and the entire design of the Experience Cylinder is built on a metaphor. It is not clear to us how a game narrative could be augmented through the cylinder, that is to say improved in a manner unique to the Experience Cylinder. Regarding our model, narrative aspects account for the parts of involvement concerning emotional attachment, which we excluded in our game prototype.

Using the physical space and integrating cooperative elements within a game has unexplored potential. We did not implement any cooperative elements directly in our prototype, but we discovered that spectators at a given play session would communicate with the player to warn him/her about incoming meteors. This dimension is expressed in the model under context, rather than inherent game structures that support social interaction through the game.

13.1 Conclusion

Each type of immersion in game literature is related to a corresponding inherent game structure that is the foundation to cause immersion or what we have specified as either a type of presence or involvement.

It separates different aspects of an experience, and unlike game immersion it takes platform into account, by distinguishing between form, content, and user.

By using the development of a game prototype as a vehicle for the investigation, we showed how to utilize the features of Unity3D to realize the possibility of presenting a 3D virtual world inside the Experience Cylinder. Additionally we explained the math involved in implementing a more advanced and precise shader, than the one we managed to implement within the scope of this thesis.

We found that smoothing and moving averages did have an effect on the controls of the game but it requires a decision between responsiveness and predictability, which one has to assess for individual purposes.

But we have identified limitations or deficiencies, which suggest future improvements. These limitations and deficiencies include, but are not limited to: projector distortion which causes a repeated curve in the top and bottom of the display, folding canvas that negatively affects the impression of objects in motion, and a top-mounted Kinect which excludes use of the existing body tracking features.

We have identified challenges and presented possible solutions. The individual solutions adequately dealt with these challenges, but the potential remains to create more general solutions, as our solutions are specific for our situation and the limiting aspects of the Experience Cylinder.

We have found that each type of immersion in game literature is related to a corresponding inherent game structure that is the foundation to cause immersion, or what we have specified as either a type of presence or involvement.

The findings from our analysis of these concepts were presented as a model, drawing from and integrating relevant concepts from virtual reality and game literature.

Our model separates different aspects of an experience, and unlike game immersion it takes platform into account, by distinguishing between form, content, and user. Through this distinction the model helped us evaluate the prototypes we built.

The model itself, however, was over-simplistic in some aspects. The linear connections between from form through immersion to presence, and from content through game structures to involvement, are the primary connections, but not necessarily the only possible connections between the concepts.

Through this we have demonstrated the ability of the Experience Cylinder to support immersive game experiences.

14 Recommendations

Our work with the Experience Cylinder has had its fair share of challenges to overcome, but not all of them were relevant to bring up in this thesis. Many issues had to do with simple hardware or software-issues or had to do with the physical construction itself.

14.1 Display Surface

Our first recommendation for future improvements to the experience cylinder: make the display surface rigid. The easiest solution would be to change the circular shape into a hexagonal shape, one side for each projector. This would ease alignment of the projectors and remove the necessity for curved projection correction but at the same time remove one of the unique aspects of the cylinder. But even keeping the cylindrical shape and just making the canvas rigid would be a great step in the right direction. That would make the more advanced shader we suggested usable.

14.2 Projector Position

We found that applying the shader had two unfortunate side effects. The ability of the shader was to move what we call healthy pixels upwards in a curved shape. This has the result of healthy pixels being moved out of visibility over the top and at the same time in the bottom move what we call unhealthy pixels into visibility. These unhealthy pixels were copies of the lowest row of pixels in the healthy part that got placed in vertical lines below their healthy counterpart. These issues occur because the shader does not fix the horizontal edges. These are still as curved as the previous images show. The shader only fixes the pixels within the distorted rectangle. The distorted rectangle is still visible so the horizontal edges are still curved.

The issue in the bottom of the display should be fixed by creating shader code that makes the unhealthy pixels turn black instead of being coloured in vertical lines. This would at the same time make the bottom edge of the display seem as a straight horizontal line as if the canvas was not curved. The top is possibly a bit more difficult. That would require making very specific healthy pixels black in order to cut away the curved shape in the top edge. But we propose a very simple solution to the problem, which is as simple as projector positioning. Hang the projectors so the top of the image hits the canvas from a perpendicular direction like in the following Illustration 14-1. Notice there is no distortion in the top, and only everything below needs to be shifted upwards. This way no healthy pixels gets shifted out of the image by the shader and no healthy pixels needs to be black.

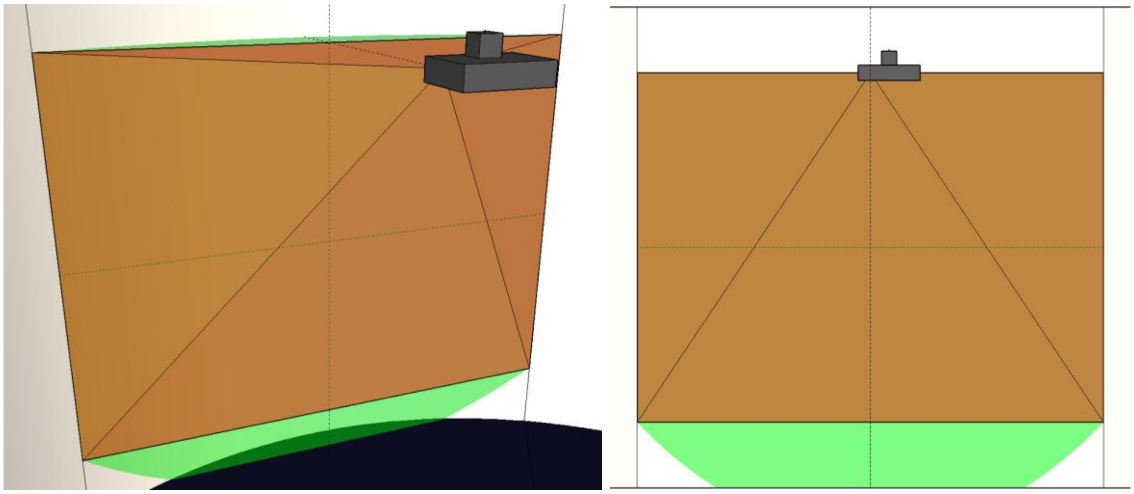


Illustration 14-1: This model shows a projector position that makes it easier to get a good result from a shader program. Notice in the image to the right that there is no distortion in the top of the display.

Additionally the Experience Cylinder would benefit from edge blending and that the image expands from the floor level up to the above-mentioned projector position. The following Illustration 14-2 proposes a projector position prepared for that. It shows two of the six projectors with an overlap needed for edge blending. But a shader for this setup requires additional features. For instance the vertical edges on the sides have to fade to black in order for the edges to merge. Furthermore the merging requires two adjacent displays to be able to render a part of each other where the overlap happens. This is possible to create in Unity3D by expanding the field of view. But the downside of this projector position is that this solution only applies the features to the applications created in Unity3D and other systems where it is possible to apply the shader and change the rendering area for each display. So when the Windows desktop is visible there would be annoying overlaps with no seamless blending. More expensive hardware solutions fix this though.

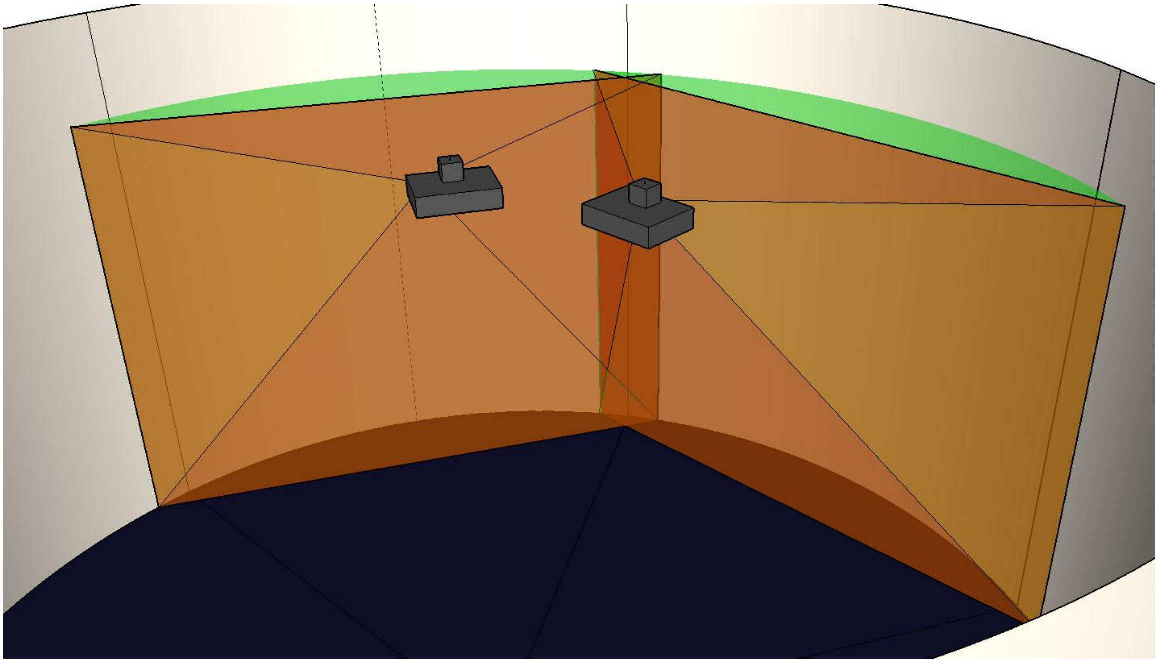


Illustration 14-2: This model shows a projector position similar to that of Illustration 14-1, but further back towards the centre of the Experience Cylinder. The position also results in an overlap with the adjacent projector in order to realize edge blending.

Furthermore a projector positioned not only in front of the centre of each display but also as close as possible to the actual centre of the experience cylinder, would result in benefits. If we implemented a shader for the vertical pixel distortion, like we did for the horizontal pixel distortion, we would fix the horizontal distortion from the centre point of view. Positioning the projectors very close to the centre point of view would possibly make it look approximately the same, possibly enough to not being able to see the horizontal pixel displacement problem after all.

14.3 Stereoscopic 3D

A short notice on stereoscopic 3D is taken additionally. Some of our examples of similar installations, see section 7.2 Similar Installations, utilised stereoscopic 3D to create an enhanced sense of depth by simulating objects appearing in front of the display surface. Without stereoscopic 3D, it is more difficult to create a feeling of depth.

This problem creates an additional issue of using the big Experience Cylinder and even the smaller CAVE to display these 3D worlds without stereoscopic 3D. For instance it looks wrong if you attempt to virtually represent small spaces in the experience cylinder. The reason is that the virtual objects cannot be detached from the display surface, preventing a convincing display of environments smaller than the physical size of the experience cylinder. The environment is displayed on the display surface of the experience cylinder and therefore seems to be six meters wide. A stereoscopic 3D solution would be able to create the perception that elements of the environment are closer than the canvas, and thereby fix the problem of entering parts of any 3D virtual environment that are narrower than the diameter of the Experience Cylinder, and should be displayed as such.

Such stereoscopic 3D solution requires either active or non-active glasses. By non-active glasses the amount of projectors needs to be doubled to twelve, but by active glasses we only need capable graphics cards and 3D-ready projectors. According to specifications, the current graphics cards and projectors of the experience cylinder already support stereoscopic 3D. The only thing we need is a software upgrade and the glasses, preferably ones with a wireless connection to the computer. We suggest RF as remote connection rather than IR controlled glasses in order to not having to ad IR emitters at the sides of the cylinder. In Unity3D it should be possible to create the additional virtual camera setup needed to create stereoscopic 3D in our game prototypes.

14.4 Kinect Position

The various issues with the Kinect could also warrant some changes. The top mounted position of the Kinect makes it easier to work with tracking on the horizontal plane, but it seems unneeded when frameworks like OpenNI includes functionality to estimate a person’s position in a room, which is available when the Kinect is mounted in its intended position. A setup involving several Kinects would also be a possible solution, combining them to give a view from all sides of the cylinder and avoiding blind angles. Getting a setup with multiple Kinects to work would require some customization of the software handling the Kinects, but could result in much more accurate tracking inside the cylinder.

14.5 Speaker Positions

As previously mentioned we decided to change the channels of the speakers, but without moving them, which means that the speakers still is positioned such that each display has a speaker centered behind it. Since it is now connected in the order of a surround sound setup with speaker 3 behind display 3 - see section 10.3.4 Sound Setup - it is a little bit off. Commercial surround sound setups usually assume the existence of a center speaker, but as the center speaker is behind display 3, it is not in the center. The center of the Experience Cylinder is between display 3 and 4. Therefore we additionally suggest rotating all satellite speakers 30° clockwise. Furthermore the subwoofer array could be positioned better. The thing is that even though low frequencies are difficult to place direction wise, it is still possible to register the origin of the subwoofer tower. In order to fix this we suggest distributing the subwoofers evenly around the Experience Cylinder.

15 Literature

- Adams, Ernest. *The Designer's Notebook: Postmodernism and the 3 Types of Immersion.* July 2004. http://www.gamasutra.com/view/feature/2118/the_designers_notebook_.php (accessed August 28th, 2012).
- Agarwal, Ritu, and Elena Karahanna. “Time Flies When You're Having Fun: Cognitive Absorption and Beliefs about Information.” *MIS Quarterly*, December 2000: 665-694.
- Andreasen, Troels, John Patrick Gallagher , Nikolaj Møbius , and Nicholas Padfield. “The Experience Cylinder, an immersive interactive platform : The Sea Stallion's voyage: a case study .” *AMBIENT - The First International Conference on Ambient Computing, Applications, Services and Technologies ThinkMind*, 2011.
- Badiqué, Eric, Marc Cavazza, Gudrun Klinker, Gordon Mair, and Tony Sweeney. “Entertainment Applications of Virtual Environments.” In *Handbook of Virtual Environments*, by Kay, M. Stanney, 1143 - 1166. Lawrence Erlbaum Associates, Inc, 2002.
- Bowman, Doug, A., and Ryan, P. McMahan. “Virtual Reality: How Much Immersion Is Enough?” *Computer*, July 2007.
- Bukvic, Ivica Ico, and Ji-Sun Kim. “µ MAX-UNITY3D INTEROPERABILITY TOOLKIT.” *Proceedings of the International Computer Music Conference.* Montreal, 2009.
- Csíkszentmihályi, Mihály, and Isabella Selega Csíkszentmihályi. *Optimal Experience: Psychological Studies of Flow in Consciousness.* Cambridge University Press, 1998.
- Ermi, Laura, and Frans Mäyrä. “Fundamental Components of the Gameplay Experience: Analysing Immersion.” *Proceedings of DiGRA, Changing Views – Worlds in Play*, 2005 .
- Esposito, Nicolas. “A Short and Simple Definition of What a Videogame Is.” *Proceedings of DiGRA Changing Views – Worlds in Play*, 2005.
- Heilig, Morton, L. Sensorama Simulator. USA Patent 3,050,870. 28 August 1962.
- Heim, Michael. “The Essence of VR.” In *The Metaphysics of Virtual Reality*, by Michael Heim, 109-128. New York: Oxford University Press, 1993.
- Hill, Ian. *Top 10 Most Immersive Worlds in Games.* 13th April 2012. http://www.gamingenthusiast.net/top-10-most-immersive-worlds-in-games/ (accessed August 30th, 2012).
- Hills, Heather , and James Hills. *Theory of Flow.* http://www.flowsocialmedia.com/theory-of-flow.html (accessed August 28th, 2012).
- Jacobsen, Jeffrey. “Game engine virtual reality with CaveUT.” *Computer*, April 2005a: 79 - 82.
- Jacobson, Jeffrey. *CaveUT2004.* 1st June 2005b. http://planetjeff.net/ut/CaveUT.html (accessed August 28th, 2012).

- Juarez, Alex, Willem Schonenberg, and Christoph Bartneck. “Implementing a low-cost CAVE system using the CryEngine2.” *Entertainment Computing*, 2010: 157 - 164.
- Juul, Jesper. “The Game, the Player, the World: Looking For A Heart of Gameness.” In *Level Up: Digital Games Research Conference Proceedings*, by Marinka Copier and Joost Raessens, 30 - 45. Utrecht: Utrecht University Press, 2003.
- Krueger, Myron, Thomas Gionfriddo, and Katrin Hinrichsen. “VIDEOPLACE—an artificial reality.” *CHI '85 Proceedings of the SIGCHI conference on Human factors in computing systems*, 1985: 35 - 40.
- Little Players. *Top 5 Most Immersive Games.* 6th May 2011. http://www.little-players.com/blog/2011/05/top-5-most-immersive-games/ (accessed August 30th, 2012).
- Livatino, S., V. Agerbech, , A. Johansen, and B. Johansen. “Designing a Virtual Reality Game for the CAVE.” *Eurographics*, 2006.
- Matthes, Olaf. “netsend~ for Max/MSP and Pure Data.” *Nullmedium.* 2005. http://www.nullmedium.de/dev/netsend~/ (accessed August 30th, 2012).
- McMahan, Alison. “Immersion, Engagement, and Presence - A Method for Analyzing 3-D Video Games.” In *The Video Game Theory Reader*, by Warren Robinett, 67 - 86. Routledge, 2003.
- Nunez, David. “A Connectionists Explanation of Presence in Virtual Environments.” the department of computer science, the Univserity of Cape Town, 2003.
- Nvidia. *Technology.* 2012. http://www.geforce.com/hardware/technology/physx/technology (accessed August 30th, 2012).
- PrimeSense. *PrimeSense - Natural Interaction.* 2012. http://www.primesense.com (accessed August 30, 2012).
- PrimeSense, Willow Garage, Side-Kick, ASUS, and AppSide. *OpenNI > Home.* November 2010. http://openni.org/ (accessed 8 28, 2012).
- Pulkki, Ville. “Generic panning tools for MAX/MSP.” *Proceedings of International Computer Music Conference.* 2000. 304 - 307.
- RUC. *Experience Lab.* 2012. http://experiencelab.ruc.dk (accessed August 30th, 2012).
- Sanford, Karl. *Smoothing Kinect Depth Frames in Real-Time.* 24th January 2012. http://www.codeproject.com/Articles/317974/KinectDepthSmoothing (accessed August 30th, 2012).
- Sheridan, Thomas B. “Musings of telepresence and virtual presence.” *Presence: Teleoperators and Virtual Environments*, 120-125.

- Shochet, Joe, and Jesse Schell. *GDC 2001: Interactive Theme Park Rides.* 3rd July 2001. http://www.gamasutra.com/view/feature/3060/gdc_2001_interactive_theme_park_.php (accessed August 28th, 2012).
- Shotton, Jamie, et al. "Real-Time Human Pose Recognition in Parts from Single Depth Images." *IEEE Computer Vision and Patteren Recognition (CVPR).* 2011.
- Slater, Mel. "A Note on Presence Terminology." *Presence Connect*, 2004.
- Soundbible. *Soundbible.* 2012. http://soundbible.com (accessed August 30th, 2012).
- Thon, Jan-Noël. "Immersion Revisited: On the Value of a Contested Concept." In *Extending Experiences Structure analysis and design and computer game player experience*, by Hanna Wirman, and Amyris Fernandez Olli Leino, 29 - 43. Lapland University Press, 2008.
- Thorlund, Steffen, Kasper Søfren, Martin Knudsholt, and Lasse Ronnenberg. "3D Modeling and Pathfinding in Java." *RUDAR.* 18th June 2009. http://rudar.ruc.dk/handle/1800/4307 (accessed August 30, 2012).
- TKK Acoustics Laboratory. "Vector base amplitude panning." *TKK Acoustics Laboratory.* 18th 1 2006. http://www.acoustics.hut.fi/research/cat/vbap/ (accessed August 30th, 2012).
- Trimble. *Trimble SketchUp.* http://www.sketchup.com/intl/en/index.html (accessed August 28th, 2012).
- Unity3D. *Camera.* 10 11 2011. http://docs.unity3d.com/Documentation/Components/class-Camera.html (accessed August 30, 2012).
- van den Ende, Jan, and Wilfred Dolfsma. "Technology push, demand pull and the shaping of technological paradigms - Patterns in the development of computing technology." *JOURNAL OF EVOLUTIONARY ECONOMICS*, 2005: 83-99.
- Weniger, S., and C. Loebbecke. "Cognitive Absorption and the Use of Hedonic IS: Literature Review and Suitability Assessment." *9th SIG IS Cognitive Research Exchange (CoRE) Workshop (pre-ICIS)*, December 2010.
- Wikipedia. *Video Game Genres.* 22th August 2012. http://en.wikipedia.org/wiki/Video_game_genre (accessed August 28th, 2012).