Department of Computer Science

A Toolkit for Virtual Reality Software Development

Investigating Challenges, Developers, and Users

Tuukka M. Takala

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 6/2017

A Toolkit for Virtual Reality Software Development

Investigating Challenges, Developers, and Users

Tuukka M. Takala

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on 27th of January 2017 at noon.

Aalto University
School of Science
Department of Computer Science

Supervising professor
Professor Tapio Takala, Aalto University, Finland

Thesis advisor
Professor Tapio Takala, Aalto University, Finland

Preliminary examiners
Professor Mark Billinghurst, University of South Australia, Australia.
Professor Petri Pulli, University of Oulu, Finland.

Opponent
Professor Ernst Kruijff, Bonn-Rhein-Sieg University of Applied Sciences, Germany

Aalto University publication series
DOCTORAL DISSERTATIONS 6/2017

ISBN 978-952-60-7245-6 (printed)
ISBN 978-952-60-7244-9 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-7244-9

Unigrafia Oy
Helsinki 2017

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Tuukka M. Takala

Name of the doctoral dissertation
A Toolkit for Virtual Reality Software Development: Investigating Challenges, Developers, and Users

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 6/2017

Field of research Virtual Reality

Manuscript submitted 13 June 2016 **Date of the defence** 27 January 2017

Permission to publish granted (date) 4 November 2016 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Possibilities of virtual reality (VR) technology have gained considerable attention recently due to technical advances in affordable head-mounted displays. Yet the use of VR technology has not become mainstream, and there still does not exist a "killer application" for VR. One reason for this situation could be the inherent difficulty of VR software development.

This thesis investigates challenges specific to VR software development, and explores methodology for such research. The thesis includes some of the earliest quantitative analysis on VR software development challenges, identifies the most severe development issues, and proposes solutions to them. This has implications on how VR software development could be eased. The analysis is based on data collected from 132 developers of VR application programs, which forms the backbone of the research.

The thesis introduces RUIS, a software toolkit for facilitating hobbyist innovation by simplifying the development of VR application programs that rely on immersive displays and spatial interaction devices. Case studies employing VR application programs created with RUIS are included, describing different ways how 3D user interfaces can affect the experience and performance of VR software users.

Methodology for benchmarking VR toolkits is presented, RUIS is contrasted with other toolkits, and multiple VR application programs created by students with RUIS are juxtaposed. The results demonstrate the importance of the chosen VR toolkit for the development process in two ways: 1) by presenting several comparisons that show how different VR toolkits can significantly affect the experienced development challenges, and 2) by highlighting the quantifiable distinctions in VR application programs created with different toolkits.

Additionally, this thesis features an extensive survey on the developers of 3DUI application programs, revealing their demographics, the software and the hardware that they use, and an overview of the 3DUI application programs that they create. The survey also points out those development challenges that particularly affect inexperienced developers, and illustrates that the reuse of high-level 3D user interface features is low. Potential solutions to these issues are proposed in the thesis.

Keywords virtual reality, 3D user interface, human-computer interaction, spatial user interaction, software toolkit, software development

ISBN (printed) 978-952-60-7245-6 **ISBN (pdf)** 978-952-60-7244-9

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2017

Pages 166 **urn** http://urn.fi/URN:ISBN:978-952-60-7244-9

Aalto-yliopisto

Tiivistelmä

Aalto-yliopisto, PL 11000, 00076 Aalto www.aalto.fi

Tekijä
Tuukka M. Takala

Väitöskirjan nimi
Ohjelmistoalusta virtuaalitodellisuussovellusten kehitykseen: Katsaus haasteisiin, ohjelmistokehittäjiin, ja käyttäjiin

Julkaisija Perustieteiden korkeakoulu

Yksikkö Tietotekniikan laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 6/2017

Tutkimusala Virtuaalitodellisuus

Käsikirjoituksen pvm 13.06.2016 **Väitöspäivä** 27.01.2017

Julkaisuluvan myöntämispäivä 04.11.2016 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

Tiivistelmä

Virtuaalitodellisuuden mahdollisuudet ovat viime aikoina saaneet huomattavasti julkisuutta johtuen teknisistä edistysaskelista edullisten virtuaalilasien saralla. Virtuaalitodellisuusteknologian käyttö ei ole kuitenkaan yleistynyt laajasti, ja vielä ei ole olemassa virtuaalitodellisuuden läpimurtosovellusta. Eräs syy tälle tilanteelle voi olla virtuaalitodellisuuden ohjelmistokehityksen vaikeus, ja siihen kuuluvat erityisongelmat.

Tässä väitöskirjatyössä tutkitaan virtuaalitodellisuuden ohjelmistokehitykselle ominaisia haasteita ja tarkastellaan aiheeseen liittyvää tutkimusmetodologiaa. Väitöskirja sisältää ensimmäisten joukossa tehtyä kvantitatiivista analyysiä virtuaalitodellisuuden ohjelmistokehityksen haasteista, identifioi kaikista vaikeimmat ongelmat, ja ehdottaa ratkaisuja niihin. Tulokset antavat viitteitä siitä miten virtuaalitodellisuuden ohjelmistokehitystä voi helpottaa. Tehty analyysi perustuu kyselytutkimukseen, johon on vastannut 132 ohjelmistokehittäjää, mikä muodostaa väitöskirjatyön kulmakiven.

Väitöskirja esittelee RUIS-ohjelmistoalustan, joka on tarkoitettu edistämään harrastelijakehittäjien innovaatioita yksinkertaistamalla immersiivistä teknologiaa käyttävien virtuaalitodellisuussovellusten kehittämistä. RUIS:lla toteutettujen sovellusten case-tutkimuksia käydään läpi, kuvaten eri tapoja, joilla kolmiulotteiset käyttöliittymät voivat vaikuttaa käyttäjien kokemuksiin ja suorituskykyyn.

Väitöskirjassa lanseerataan menetelmiä virtuaalitodellisuuden ohjelmistoalustojen vertailemiseksi, RUIS:ia tarkastellaan muihin ohjelmistoalustoihin nähden, ja useita opiskelijoiden RUIS:lla kehittämiä virtuaalitodellisuussovelluksia verrataan keskenään. Saadut tulokset havainnollistavat ohjelmistoalustojen merkityksen virtuaalitodellisuussovellusten kehitysprosessille kahdella tapaa: 1) esittämällä kuinka käytetty ohjelmistoalusta voi vaikuttaa tilastollisesti merkitsevästi kehitystyössä koettuihin haasteisiin, ja 2) korostamalla kvantitatiivisia eroja virtuaalitodellisuussovelluksissa, jotka on toteutettu eri ohjelmistoalustoilla.

Lisäksi väitöskirjatyö sisältää laajan selvityksen kolmiulotteisia käyttöliittymiä hyödyntävien sovellusten kehittäjistä, paljastaen heidän taustansa, käytetyt ohjelmistot sekä laitteet, ja yleisnäkymän kehitetyistä sovelluksista. Selvitys tuo ilmi erityisesti kokemattomia ohjelmistokehittäjiä vaivaavat haasteet, sekä osoittaa että korkean abstraktiotason kolmiulotteisten käyttöliittymäkomponenttien uudelleenkäyttö on harvinaista. Näihin ongelmiin ehdotetaan mahdollisia ratkaisuja.

Avainsanat virtuaalitodellisuus, kolmiulotteiset käyttöliittymät, HCI, kehollliset käyttöliittymät, ohjelmistoalustat, ohjelmistokehitys

ISBN (painettu) 978-952-60-7245-6 **ISBN (pdf)** 978-952-60-7244-9

ISSN-L 1799-4934 **ISSN (painettu)** 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Helsinki **Painopaikka** Helsinki **Vuosi** 2017

Sivumäärä 166 **urn** http://urn.fi/URN:ISBN:978-952-60-7244-9

Acknowledgements

The work presented in this thesis is done between 2010 and 2016 in Aalto University. I wish to thank my advisor, Professor Tapio Takala, and colleagues Klaus Förger, Meeri Mäkäräinen, and Roberto Pugliese. I am also thankful to Lauri Savioja, Tapio Lokki, Perttu Hämäläinen, Lauri Malmi, Jari Takatalo, Päivi Rauhamaa, Seppo Äyräväinen, Robert Albrecht, Jari Kätsyri, Yu Shen, Mikael Matveinen, Lauri Lehtonen, and Heikki Heiskanen.

I want to express special thanks to my family – Seija, Heimo, Riikka, and Jukka – without whom it would not have been possible to complete this thesis.

My work has been funded by Finnish Doctoral Program in User-Centered Information Technology (UCIT) and The Research Foundation of Helsinki University of Technology. Additionally, I have been supported by Jenny and Antti Wihuri Foundation, Emil Aaltonen Foundation, and Nokia Foundation.

Helsinki, 27th of October 2016
Tuukka Mikael Takala

Contents

List of Abbreviations and Symbols

2D	two-dimensional
3D	three-dimensional
3DUI	3D user interface
6DOF	six degrees of freedom
ANOVA	analysis of variance
AR	augmented reality
CA	correspondence analysis
CAVE	cave automatic virtual environment
HCI	human-computer interaction
HMD	head-mounted display
LED	light-emitting diode
MBC	Mecanim blended character
MR	mixed reality
RUIS	reality-based user interface system
SAM	self-assessment manikin
SEARIS	software engineering and architectures for realtime interactive systems
VR	virtual reality
WIMP	window, icon, menu, and pointing device

List of Publications

This doctoral thesis consists of a summary and of the following publications which are referred to in the text by their numerals:

P1. Takala, Tuukka. 2014. RUIS – A Toolkit for Developing Virtual Reality Applications with Spatial Interaction. In: Proceedings of the 2nd Symposium on Spatial User Interaction (SUI'14), Honolulu, USA, October 4–5, 2014. ACM. Pages 94–103. DOI: 10.1145/2659766.2659774.

P2. Takala, Tuukka; Rauhamaa, Päivi; Takala, Tapio. 2012. Survey of 3DUI Applications and Development Challenges. In: Proceedings of the Symposium on 3D User Interfaces (3DUI 2012), Orange County, USA, March 4–5, 2012. IEEE. Pages 89–96. DOI: 10.1109/3DUI.2012.6184190.

P3. Takala, Tuukka; Malmi, Lauri; Pugliese, Roberto; Takala, Tapio. 2016. Empowering Students to Create Better Virtual Reality Applications: Longitudinal Study of a VR Capstone Course. Informatics in Education, Volume 15, No. 2, 2016. Pages 287–317. DOI: 10.15388/infedu.2016.15.

P4. Takala, Tuukka; Hämäläinen, Perttu; Matveinen, Mikael; Simonen, Taru; Takatalo, Jari. 2015. Enhancing Spatial Perception and User Experience in Video Games with Volumetric Shadows. In: T. Wyeld, P. Calder & H. Shen (Eds.), Computer-Human Interaction. Cognitive Effects of Spatial Interaction, Learning, and Ability. Springer LNCS. Pages 91-113. DOI: 10.1007/978-3-319-16940-8_5. ISBN: 978-3-319-16939-2.

P5. Takala, Tuukka; Mäkäräinen Meeri; Hämäläinen, Perttu. 2013. Immersive 3D modeling with Blender and off-the-shelf hardware. In: Proceedings of the Symposium on 3D User Interfaces (3DUI 2013), Orlando, USA, March 16–17, 2013. IEEE. Pages 191–192. DOI: 10.1109/3DUI.2013.6550243.

P6. Takala, Tuukka; Pugliese Roberto; Rauhamaa, Päivi; Takala, Tapio. 2011. Reality-based User Interface System (RUIS). In: Proceedings of the Symposium on 3D User Interfaces (3DUI 2011), Singapore, March 19–20, 2011. IEEE. Pages 141–142. DOI: 10.1109/3DUI.2011.5759245.

Author's Contribution

Publication [P1]: RUIS – A Toolkit for Developing Virtual Reality Applications with Spatial Interaction

The author created the requirements and designed the software architecture of the presented toolkits, and implemented major portions of them. The author wrote the entirety of the paper.

Publication [P2]: Survey of 3DUI Applications and Development Challenges

The author wrote 90% of the paper, created the associated 3DUI questionnaire, gathered participants online, and performed statistical analysis on the questionnaire results. The author also devised methodology for studying development challenges and benchmarks for comparing virtual reality toolkits. To the best of our knowledge, this is the first paper to present and analyze quantitative data on 3DUI development challenges.

Publication [P3]: Empowering Students to Create Better Virtual Reality Applications: Longitudinal Study of a VR Capstone Course

The author wrote 90% of the paper, was involved in designing and improving the VR course in question, as well as acted as the head teaching assistant of the VR course for the five years covered in the study. Additionally, the author conducted 90% of the statistical analysis on the data collected from students via questionnaires and observations.

Publication [P4]: Enhancing Spatial Perception and User Experience in Video Games with Volumetric Shadows

The author coordinated the work, wrote half of the paper, invented the use of volumetric shadows as spatial cues, was involved in the design and implementation of the game used in the study, conducted third of the user studies, and performed statistical analysis on gameplay metrics. The original paper was published in proceedings of OzCHI 2013, and this thesis includes a version that we revised for a book upon its editor's request.

Publication [P5]: Immersive 3D modeling with Blender and off-the-shelf hardware

The author wrote 95% of the paper, invented and implemented the hybrid 2D/3D user interface, conducted the user study, and performed statistical analysis on the results. The publication won an award in IEEE 3DUI Contest, which limited the size to 2 pages, and required creating a video of the solution (youtu.be/mF0ioY7ctkM). The results have not been published elsewhere.

Publication [P6]: Reality-based User Interface System (RUIS)

The author wrote 80% of the paper, co-designed the user study's puzzle game and implemented 70% of it, conducted the user study together with Roberto Pugliese, and performed statistical analysis on the results. The publication participated in IEEE 3DUI Contest, which limited the size to 2 pages, and required creating a video of the solution (youtu.be/0DJhH1MLp1k). The results have not been published elsewhere.

1. Introduction

Virtual reality (VR) has recently sprung to public conscious due to advances in immersive technology and large investments by industrial giants like Facebook, Google, Microsoft, Samsung, and Sony, all of whom are developing VR and augmented reality (AR) products [1]. The VR field is rapidly evolving and a multitude of affordable VR peripheral devices have been introduced via successful Kickstarter campaigns, most famously the Oculus Rift head-mounted display (HMD) in 2012. Other campaigns that reached their crowdfunding target include Sixense STEM and Control VR controllers, Virtuix Omni and Cyberith Virtualizer treadmills, Avegant Glyph and CastAR HMDs, ANTVR and DIYVR developer kits, and Technolust VR computer game.

In total over 100,000 Oculus Rift development kits have been sold [2], each unit costing around $300. With the advent of inexpensive VR technology, hobbyist VR developers have become very active; over 350 Oculus Rift demos have been created [3].

Entertainment applications represent a majority of the current VR software that is being developed by VR enthusiasts: many VR computer games are under development and there are several entities trying to popularize 360 degree video for HMDs, including companies like Jaunt and NextVR which together have received over $40 million in investments.

Despite the public enthusiasm and considerable investments in VR, a "killer application" of consumer VR is yet to be seen [4]. If such an application is not discovered, it could very well mean that this latest VR boom will end up in disappointment, as has happened in the past.

The motivation for this thesis is as follows: firstly, establishing a more clear understanding of challenges in VR software development can help in building better VR toolkits, thus easing the development process and perhaps even facilitating the discovery of the killer application of consumer VR. Secondly, addressing the challenges in VR software development makes it easier for hobbyist VR developers to get started, which could empower a large number of VR enthusiasts who otherwise might not get involved. Developer *empowerment* should be one of the most important principles that guide those who create tools for developers.

Consumer VR is taking its first steps to reach major audiences, VR as a media is advancing, and hardware manufacturers are eager to find and support popular applications. We argue that it is at this point that hobbyist developers can have the biggest impact with their innovations.

1.1 Scope of the Thesis

This thesis deals with development challenges, developers, and users of *VR application programs*. Specifically the focus is in VR application programs that employ *3D user interfaces (3DUI)* and are meant to be used via immersive displays (e.g. HMDs, stereo 3D projection walls) and 3D input devices such as six-degrees-of-freedom (6DOF) controllers. Virtual worlds such as Second Life that are normally used with a mouse and keyboard interface are out of the scope of the research at hand.

Bowman et al. [5] define 3DUI as an user interface that involves 3D interaction, i.e. interaction that occurs in spatial 3D context. In VR context 3DUIs often differ from traditional 2D interfaces that rely on the so-called WIMP (window, icon, menu, and pointing device) style of interaction. Immersive VR and 3DUIs are innately related and the concept of 3DUIs is commonly used by VR researchers. For VR application programs 3DUIs offer a natural way to interact with virtual environments, since VR often emulates reality that arguably entails the most tangible 3DUI in existence.

Publications [P1–P3] study challenges that affect VR software development, linking this thesis with the field of software engineering. Publication [P2] presents methodology for measuring VR software development challenges, and publication [P3] focuses on teaching VR software development. Furthermore, publication [P2] is a survey that charts the contemporary 3DUI developer landscape, their demographics and the hardware and software they used prior to 2012 and the arrival of Oculus Rift HMD. A vast majority of the developers examined in publications [P1–P3] – and thus the subject population of the whole thesis – is from western countries. Very few participants were from Africa, Asia, and South America, which could limit the applicability of the developer related results.

VR toolkits that are intended to be used in creation of VR computer applications also fall within the scope of this thesis. A VR toolkit called Reality-based User Interface System (RUIS) is introduced [P1, P6], which attempts to solve some of the investigated development challenges. Two methods of benchmarking different toolkits are proposed in publication [P2].

Finally, users of VR application programs are studied in publications [P4–P6], where user experience and task performance is evaluated. The two VR application programs described in [P5] and [P6] were developed for IEEE 3DUI Contest that has been organized annually since 2010 as part of the IEEE 3DUI Symposium. The 3DUI for Blender [P5] won the “Best Low-Cost, Minimally-Intrusive Solution” award and the 3DUI puzzle [P6] placed 4th out of 8 participants.

N.B.: In publications [P1] and [P3] VR application programs are referred as *VR applications*, and in publication [P2] the term *3DUI application* is used to describe an application program that employs a 3DUI. This thesis summary purposely avoids those terms, so that their meaning would not be confused with application domains, i.e. the use cases for VR and 3DUIs. The price of avoiding this confusion is having to rely on somewhat uncommon terms *VR application program* and *3DUI application program*.

1.2 Research Questions

VR software development contains many specific challenges [6–8] that hinder the design and implementation of VR application programs. If these challenges are not addressed properly, this could negatively affect the usefulness and adoption of VR technology, and possibly have consequences in related fields such as AR and mixed reality (MR), ubiquitous and pervasive computing [7].

This thesis poses the following research questions: *1) What are the most severe challenges in VR software development?* Previous research has identified different challenges and provided anecdotal evidence about them, while this work attempts to measure the severity of these challenges. *2) How to facilitate VR software development and address its challenges?* The role of VR software toolkits in the development process is particularly inspected. *3) What benefit does immersive VR offer to software users in contrast to traditional application programs?* This fundamental question regarding the rationale of using VR technology is also touched on in the thesis.

The immediate goal of the thesis is to present an analysis on VR software development challenges and provide guidelines for creating VR toolkits that take those challenges into account. Hopefully this research has impact on future VR toolkits that aim to ease the implementation of VR application programs and make VR software development more accessible for hobbyist developers.

The research presented in this thesis could be of interest to specialists of AR and MR, because VR has much in common with those technologies – including 3DUIs and the utilization of similar software tools for development.

1.3 Research Methods

This thesis represents open-ended, constructive research [9]. First – after a literature review – a VR toolkit called RUIS was created [P1, P6]. This was followed by the creation of several VR application programs [P4–P6], which were used as part of case studies that extracted information about the VR user experiences. Standard usability research methods and statistical analysis were applied in the case studies, which involved tasks performed by study participants whose feedback and performance in the studies was evaluated.

An extensive questionnaire was created to gather data about 3DUI application programs [P2], their developers and the experiences of the developers. Between 2011 and 2016 a total of 167 participants answered the questionnaire. Most of these were students of a virtual reality course organized in Aalto University, where the author of this thesis acted as the head teaching assistant [P3]. The questionnaire participants' answers yielded quantitative and qualitative data about development challenges.

The gathered data regarding VR application program developers and users was analyzed for statistically significant results with statistical tests appropriate for each occasion: Kruskal-Wallis test, Wilcoxon rank-sum test, Wilcoxon signed-rank test, Friedman test, analysis of variance (ANOVA), and Fisher's exact test. When comparing multiple groups, a post-test with either Bonferroni or Tukey-Kramer correction was applied, depending on the case. In publica-

tion [P4] qualitative data was explored with correspondence analysis. MATLAB software was used to do most of the statistical analysis, and the rest was done with SPSS Statistics software.

1.4 Thesis Structure

Themes of this thesis and their relationships are illustrated in Figure 1: the overarching theme is VR, the focus being in VR toolkits, VR developers, and development challenges. A brief introduction to these topics is given in Chapter 2, alongside with the related themes of software engineering, teaching VR, and user-led innovation.

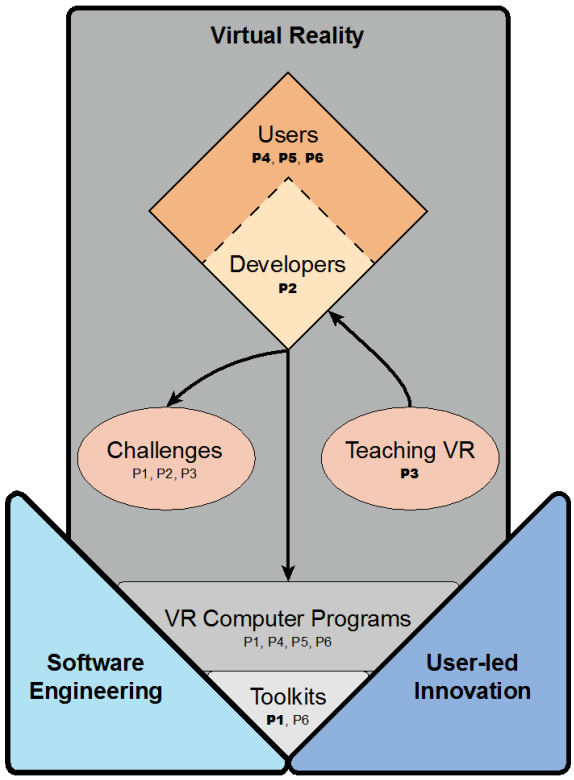


Figure 1. Themes of the thesis, how they are related to each other, and how publications P1–P6 exhibit them. Bold font indicates the main theme of a publication, while regular font expresses additional, minor themes.

Figure 1 depicts how VR toolkits exist conceptually at the intersection of software engineering and user-led innovation, and are used by VR developers to create VR application programs. Figure 1 also illustrates how VR users are a superset of VR developers, who experience various development challenges and can also be subjected to teaching VR development.

Chapter 3 presents the RUIS toolkit, which is our way of addressing research question 2. Chapter 4 concentrates on developers of VR application programs, the toolkits that they use, and their challenges. Research question 1 is an-

swered in Section 4.2. Chapter 5 focuses on users of VR application programs and introduces case studies that relate to research question 3. In Chapter 6 we discuss our results, paying special attention to the most severe development challenges and their possible solutions, thus further addressing research question 2. Lastly, Chapter 7 summarizes our research and its contributions.

2. Related Work

This chapter introduces the topics relevant to this thesis and presents a selection of related literature that can be read to gain a deeper understanding of the topics.

2.1 Virtual Reality

Several definitions for virtual reality exist, and here we present one by Sherman and Craig, who identified four key elements of VR: a virtual world, immersion, sensory feedback, and interactivity. Using these elements they defined VR as "*a medium composed of interactive computer simulations that sense the participant's position and actions and replace or augment the feedback to one or more senses, giving the feeling of being mentally immersed or present in the simulation (a* virtual *world)*". [10]

VR systems primarily convey output to visual and auditory senses, and modern VR can offer quite realistic audiovisual experiences. Visual stimulation is usually delivered via a HMD, CAVE [11], or stereo 3D projection walls, whereas headphones or speaker arrays are employed for auditory stimulation. VR hardware capable of providing haptic feedback is also somewhat common, but anything beyond vibrotactile stimulus is expensive and limited in fidelity due to technological challenges related to haptics. VR interfaces that provide olfactory feedback are quite rare, and even rarer are interfaces with gustatory feedback [10].

As for input technology, VR relies heavily on motion tracking, which is employed to track the user of the VR application program and to adjust the feedback from the virtual world accordingly. In many VR systems the user utilizes hand-held 3D input devices that have buttons for interaction purposes. A more detailed description of VR input and output technologies is given by Sherman and Craig's textbook [10].

VR technology was first employed for data visualization and training purposes [10], and later on VR has spread to design, telepresence, collaborative working, and entertainment domains [12]. Traditionally the high cost of VR hardware has restricted the use of VR mostly to research laboratories and big companies. LaViola noted that video game companies have been introducing VR and spatial 3D interaction to consumers since the late 1980s with peripherals such as Mattel Power Glove, Sega 3-D Glasses, Sony EyeToy, and Nintendo Wii

Remote [13]. Currently we are witnessing the biggest push for consumer VR with the release of Oculus Rift, HTC Vive, Samsung Gear VR, and other affordable HMDs. As display technology advances further in the coming years, we can expect a similar situation with AR devices.

First VR research dates back to 1960s, to Sutherland's work with the first motion-tracked HMD [14]. VR research usually concerns VR hardware, software, task performance, and psychology. Particularly the latter two topics are often explored via user studies, where statistical tests are employed to accept or reject hypotheses. Andujar and Brunet [15] discussed the application of user studies in the field of VR and some of the issues related to their use in VR research. They noted that VR user study design is challenging due to the large number of factors involved (including the multitude of hardware configurations), many of which need to be fixed arbitrarily in order to keep number of experiment conditions feasible. Andujar and Brunet also argued that "*a large body of findings in VR and 3DUI only apply to very specific conditions, and it is unclear if the results can be generalized to other contexts.*" [15] The same notion was also mentioned by Wingrave and LaViola [7].

2.2 Software Engineering

Because this thesis deals with developers, users, development tools, and development difficulties, it is inherently connected with software engineering research. Below we present some of the previous VR and 3DUI literature that is tightly coupled with field of software engineering. For example, a large body of such literature has emerged via the Software Engineering and Architectures for Realtime Interactive Systems Working Group (SEARIS), which has organized the annual SEARIS Workshop since 2008 [16].

Figure 2 represents the different conceptual layers of software engineering, as described by Pressman and Maxim's comprehensive book on the topic [17]: focus in software quality forms the foundation of software engineering. Processes establish how methods are applied and software projects are managed. Methods supply technical descriptions on how to perform different tasks of software engineering, such as designing and building software. Tools include computer programs used for planning and implementing software, as well as otherwise supporting the process.

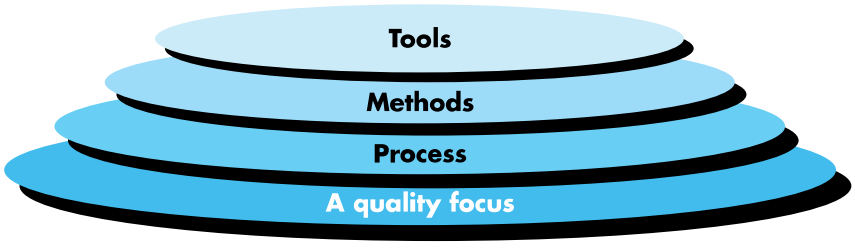


Figure 2. Software engineering layers. Reprinted from "Software Engineering: A Practitioner's Approach" by R. S. Pressman and B. R. Maxim, 2015, p. 16. Copyright 2015 by McGraw-Hill Education. Reprinted with permission.

Much of VR software engineering literature deals with software tools and toolkits; dozens of VR toolkits have been released. Most VR toolkits are contemporary, because it takes effort to upgrade a toolkit to conform to the constantly evolving hardware and graphics libraries. Rare examples of long-lasting VR toolkits that have received updates for over a decade include VRPN [18], VR Juggler [19], and WorldViz Vizard [20].

Methods for VR development are more long-lasting as they are abstract, but it is unclear how often they are applied by VR practitioners. In 1998 Kim et al. released an early paper on a more structured software engineering approach to VR development [21], presenting methods for creating specifications for form, function, and behavior of VR application programs. Other examples of VR software engineering methods include a VR user interface design model by Tanriverdi and Jacob [22] and VR design methodology by Kaur [23], whose work was further discussed by Fencott [24].

Mattioli et al. [25] described an agile development process for VR, which takes elements from extreme programming and Scrum. As far as we are aware, no other software engineering processes have been proposed for VR development. This could be because the existing software engineering processes are adequate for VR development, or perhaps because VR software projects are rarely so large that the role of the process would become emphasized.

2.2.1 Challenges in VR Software Design and Implementation

Already in 1991 Green and Jacob [6] pointed out several issues that affect the design and implementation of VR application programs. Subsequently there has been some papers dealing with these issues: Myers [26] listed many human-computer interaction (HCI) design and implementation difficulties, most of which apply directly to VR development. Steed [8] discussed issues with VR software and hardware reuse, as well as their typically short lifespan. Research published at SEARIS workshops often relates to VR development challenges [27–29].

Wingrave and LaViola [7] concluded in their literature review that despite decades of advances in VR technology, little has changed in VR software design and implementation, which is still riddled with challenges. They presented a comprehensive list with 67 VR design and implementation issues, based on their experience, developer interviews, and design artefacts. The issues were categorized into 11 themes (indicated in bold):

1. **Multiple Varied Skills** such as programming, interface design, content creation, and engineering are required in VR design and implementation, which is a demanding cross-disciplinary process.
2. **Human Experience and Perception** is integral to VR, and developers need to understand user expectations, psychology, nausea, differences between users, etc.
3. **Content**: acquiring and incorporating assets such as 3D models and audio in high quality has challenges.

4. **Design Knowledge**: VR design is intricate, and related standards and guidelines have much room for improvement.
5. **Iterative Prototyping** is often necessary and has a negative impact on the design and implementation process.
6. **Representation and Reuse**: it is easier to build new systems than to reuse existing ones, and the application of formal models is problematic.
7. **Complex, Chaotic, and Difficult** is the nature of VR application program development due to the many different components, hidden connections, and challenging solutions involved.
8. **Real-Time Operation**: interactive VR application programs require an efficient implementation and the use of parallel and threaded computing architectures, which complicates development.
9. **Callbacks and Events** are commonly used in VR development, but they organize system complexity insufficiently and cause challenges for system maintenance and reusability.
10. **Hardware**: it is difficult to determine and obtain the best VR hardware configuration, and there is a lack of hardware standards.
11. **Tools and Community**: VR developers are divided by the tools that they use, which hinders sharing of implementations and ideas.

Wingrave and LaViola argued that finding solutions to the aforementioned issues will improve VR application programs and possibly alleviate similar difficulties in related fields of AR and MR. [7]

To the best of our knowledge, no quantitative data has been presented about challenges in VR or 3DUI development prior to our survey [P2] that was published in 2012. A certain level of justification for the survey's approach of examining 3DUI development challenges is provided by research of Lahtinen et al. [30] and Tan et al. [31], who studied the difficulties of novice programmers by employing a multitude of Likert statements rated by study participants. Their statements regarding practical difficulties in learning programming are similar to our development difficulty statements in publication [P2].

2.3 User-led Innovation

In the past 10 years consumer versions of immersive display technology and motion controllers – essential components of VR hardware – have become very affordable; examples include 3D televisions and game peripherals such as PlayStation Move controllers.

In 2010, partly inspired by Johnny Chung Lee's work with Nintendo Wii Remotes [32], it occurred to the author of this thesis that the wide availability of immersive technology would facilitate hobbyist innovations in the field of VR, which in turn could advance VR research and mainstream adoption. This

realization heavily influenced the creation of the RUIS toolkit, whose purpose since its inception has been to ease VR development and unleash the innovation potential of hobbyist VR developers [P6]. With this being the background motivation for the work at hand, this thesis includes a user-led innovation perspective on VR. In this section we introduce basic user-led innovation concepts, examine VR user innovation from the past 30 years, and finally utilize user-led innovation literature to characterize recent, drastic advances in the field of consumer VR.

Von Hippel wrote a comprehensive overview on user-led innovation, drawing from 30 years of research in the field [33]. He emphasized that it is not only producer companies that innovate products, but also product users who tend to share and freely reveal their innovations. In the context of user-led innovation, Von Hippel categorized users into individual consumers, communities, and firms [33], who have increasingly more options to create and modify products to meet their needs.

Von Hippel et al. presented a three-phased consumer innovation paradigm that outlines how users can act as pioneers and create functionally novel products where market supply does not exist yet: [34]

I. Users develop new products for themselves.
II. Other users evaluate and reject, or copy and improve.
III. Producer companies enter when market potential is clear.

This paradigm is further illustrated by Figure 3, which depicts sequences of innovation activities carried out by users and producer companies, and how the two parties can interact with each other. Producer companies commonly employ a traditional innovation process, depicted in the lower part of Figure 3. Conversely, users often innovate before producers, particularly in cases where existing market does not cater to the users' needs. If the user innovator chooses to share their designs, innovation diffusion becomes possible: the innovation can spread among other users, who may continue to improve the original design, which in turn could even influence producer companies. [35]

While some papers exist on how VR could be harnessed to facilitate innovation within companies [36], we have not come across any research that would examine VR using existing user-led innovation literature. The only directly related piece of research is a multiple case study by Whyte, where firms in the construction sector were considered as user-innovators of virtual reality technology [37]. Whyte's study mainly focused on how VR technology was adapted in this specific sector, and it did not show whether that innovation had spread to other sectors of VR.

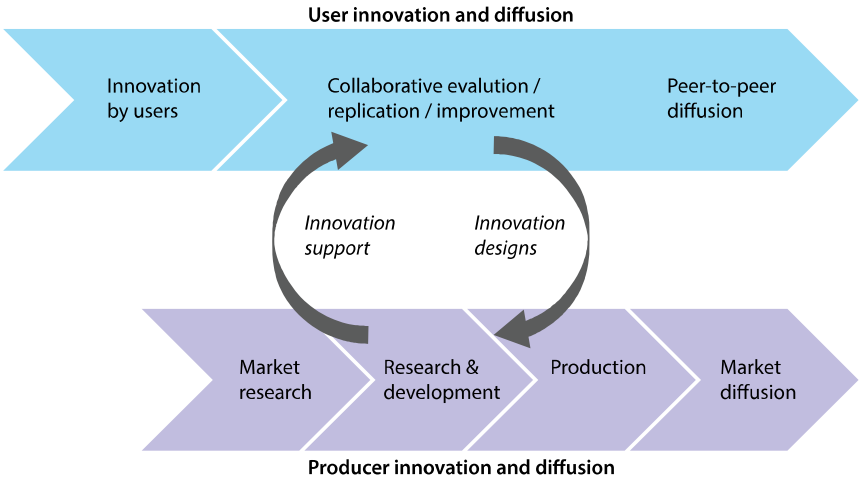


Figure 3. The user and producer innovation and diffusion paradigms. Figure adapted from "The user innovation paradigm: impacts on markets and welfare" by Gambardella et al., 2015 [35].

2.3.1 Early Virtual Reality User Innovation

Video game history has been shaped by hobbyist innovation [38], and the idea of user innovators' potential for advancing the related field of VR is not new: in 1991 Pausch remarked that the cost of hardware has limited the research on VR. He suggested that the best way to speed up breakthroughs in VR is to provide low cost versions of the equipment so that more people can use the technology. [39]

The growing interest in the possibilities of VR and more affordable hardware such as Mattel Power Glove led to an active "homebrew" VR scene in 1990s [40]. Throughout the decade several books for hobbyists were published on how to build affordable VR systems using personal computers and off-the-shelf peripherals [41–43].

In 1993 Pimentel and Teixeira pointed out the innovation happening at grass-roots level, listing several small VR start-ups of the era [44]. In the same year Weber anticipated a further rise in VR user innovation: "*with virtual reality, these living room inventors, garage scientists, gadget tinkerers, and hobbyist-entrepreneurs are resurfacing*." He also emphasized that for homebrew VR research and development to truly surge, the cost of hardware should come down from the $2500-5000 price range, which at the time was the cost of a low fidelity desktop VR setup. [45]

Despite the increased appeal of VR and grass-roots innovation in the 1990s, efforts to create consumer VR markets failed because the available technology could not meet the inflated expectations of the public, and consumer VR fell into a "trough of disillusionment" [10]. Factors contributing to this included nausea and discomfort experienced by users, which were common with the more affordable HMDs of the era [46].

The release of Nintendo Wii console and its Wii Remote controller in 2006 gave new momentum to the proponents of affordable VR. Several research

papers were published about using the Wii Remote as an VR interaction device [32, 47–49]. Microsoft's Kinect from 2010 was also met with similar enthusiasm by hobbyists and researchers [50]. The wide adoption of Kinect by hobbyists can be partly attributed to Johnny Chung Lee, who had been part of Kinect development team and who issued a $3000 software driver hack bounty after the release of the device [51].

In 2012 – just before Oculus Rift – Pausch's idea of advancing VR research via affordable hardware was echoed by Basu et al. [52] and other researchers at University of Georgia, who emphasized the importance of affordable, off-the-shelf components [53] and noted their potential for user innovation: "*VR systems developed with off-the-shelf devices can be reproduced more easily and cheaply, potentially leading to larger deployment of VR applications and further innovation.*" [54]

2.3.2 Virtual Reality User Innovation in the Oculus Rift Era

The three-phased consumer innovation paradigm presented above fits strikingly well with the backstory behind the current consumer VR boom: founded in 2007, the MTBS online forum [55] was a popular hub for VR hobbyists to share their experiences about creating VR hardware for themselves [56] (phase I). Since 2009 the Oculus VR founder Palmer Luckey – then a VR hobbyist – had frequented the forum, where he actively collaborated with other forum members while developing different HMD prototypes [57] (phase II). In 2012 Luckey founded Oculus VR, after leaving his job as a laboratory engineer at University of Southern California Institute for Creative Technologies [40], where he had been part of the team developing the mobile FOV2GO HMD [58, 59]. Impressed by the potential of Luckey's HMD prototype, Oculus VR received seed funding from a co-founder [60] and public endorsements from renowned game industry veterans, leading to a successful Kickstarter campaign in August 2012 [61]. Witnessing the enthusiasm generated by the Oculus Rift HMD, big companies like Sony, Samsung, and HTC sensed the market potential and followed up with their own HMD products (phase III).

Luckey's journey from a VR hobbyist to the founder of Oculus VR is also a case of a lead user becoming a producer, a possibility mentioned by Von Hippel [33]. It should be noted that the ingenuity in the Oculus Rift HMD prototype relied strongly on earlier work by researchers and practitioners, which often is the case with innovations, user-led or otherwise.

Today user innovation can be witnessed in the many VR meetups that are regularly organized around the world, where VR hobbyists and start-ups actively participate. In these meetups people share ideas and socialize, while developers receive feedback and enthusiasts get to try the latest applications. [62]

Big companies have also slowly started to embrace user innovation in VR and related fields: in 2011 Sony made PlayStation Move controllers usable for PC developers via their Move.me software. This was followed by Microsoft doing the same to Kinect with the official Kinect SDK released later that year. Currently companies like Oculus VR, HTC, and Valve actively support inde-

pendent developers by providing hardware and in some cases even funding, so that developers can create VR content [63]. Oculus VR and Leap Motion have also been organizing “VR jams”, where small development teams compete for prize money.

Baldwin and von Hippel [64] discussed the three models of innovation, which appear in the user innovation literature: single user, producer, and open collaborative innovation. In the context of VR, these innovation actors can be illustrated with the following present-day examples: individual VR developers who share their innovations are single user innovators, Oculus VR and Sony are producer innovators, and the parties working with Open Source Virtual Reality (OSVR) are open collaborative innovators.

Baldwin and von Hippel also introduced hybrid innovation models where the aforementioned actors benefit from each other. As an example they mentioned game engines, which are created by producer firms, and expanded upon by individual developers or groups of developers [64]. Currently the most notable game engines with large user communities are Unity 3D, Unreal Engine, and CryEngine. Each of them has recently added native support for HMDs and spatial 3D input devices.

2.4 Virtual Reality Toolkits

The concept of toolkits is found in user-led innovation [65], software engineering [17], and VR literature. The purpose of a VR toolkit is to provide reusable components that can be utilized to create VR application programs, avoid building everything from scratch, and reduce the amount of low-level programming. A VR toolkit could for example include functionality for display management, distributed rendering, or handling input devices. Notable examples of VR toolkits include an early yet comprehensive MR Toolkit by Shaw et al. [66], Carlsson and Hagsand’s long-lived DIVE [67] that went through multiple iterations, and the widely used VR Juggler by Bierbaum et al. [19].

In this thesis we use the term VR toolkit loosely, to include everything from auxiliary VR programming libraries to full VR development environments that combine programming, content management, and execution in a single software suite. In VR literature the term VR toolkit is often interchangeable with the similar terms VR framework, platform, tool, and development environment.

Only a handful of publications examine VR toolkits and their qualities generally: in 1993 Shaw et al. [66] discussed the need for VR toolkits and presented software engineering related VR toolkit requirements that are relevant even today: 1) portability of VR application programs, 2) support for a wide range of input and output devices, 3) VR application program independence from room geometry and device configurations, and 4) a flexible development environment for VR application programs.

Bierbaum and Just [68] introduced three primary requirements for a VR toolkit: ability to create high performance VR application programs, flexible development environment, and ease of use. They noted that these require-

ments often conflict, requiring compromises from the VR toolkit creator. Bierbaum and Just also presented fifteen capabilities of VR toolkits, ranging from cross-platform development to toolkit extensibility. The usefulness of these capabilities depends on the VR application program that is to be constructed with the toolkit.

Taylor et al. [69] discussed desirable and undesirable features of VR toolkits, based on 20 years of development experience. Steed's long experience with VR development led him to list several abstractions that can be useful features in a VR toolkit [8]. Varcholik et al. [70] presented multiple requirements for a research-oriented VR toolkit, and categorized them as high-level non-functional requirements, primary components necessary for basic research, and secondary components essential to pursue long-term research efforts.

Within the realm of user-led innovation research, Von Hippel lists five important attributes of a high-quality toolkit for user innovation: [33]

1. It will enable users to carry out complete cycles of trial-and-error learning.
2. It will offer users a solution space that encompasses the designs they want to create.
3. It will be user friendly in the sense of being operable with little specialized training.
4. It will contain libraries of commonly used modules that users can incorporate into custom designs.
5. It will ensure that custom products and services designed by users will be producible on a manufacturer's production equipment without modification by the manufacturer.

Attributes 1-4 are abstract yet fitting requirements for a good VR toolkit. The reproducibility attribute (5) is trivial in case of application programs, but applies well to VR hardware. One example of a pathway where user innovation in VR hardware could influence a manufacturer's production are the small tracking sensors of Valve's Lighthouse tracking system, which will be made available for users to embed in their own VR props and devices [71].

Dozens of VR toolkits as well as papers introducing them exist. There are several commercial toolkits, while a large portion of VR toolkits are created by academics. This is interesting when taking into account Wingrave and LaViola's perception that "*academics are not rewarded for building and maintaining tools. […] As such, the resources available to advance the state of the art are limited.*" [7]

Steed [8] reasoned that so many VR toolkits exist because VR software is a large domain with a multitude of different applications, each with their own requirements that can vary from efficient parallel computing to peer-to-peer networking. One VR toolkit can cover only a comparatively small area of that domain. Moreover, programming languages, libraries, and hardware keep

changing over the years, which increases the temptation of making a fresh start by creating a new VR toolkit instead of extending the existing, often complex toolkits.

Steed also explained why their laboratory had used over 40 different VR software systems in a time span of 15 years: often the chosen toolkit offered a good fit in some regard, but was later retired because it was visually outdated, retired by its author, lacked features, etc. [8]

Because a plethora of VR toolkits exist, comparison of toolkits is important so that a developer can choose a particular VR toolkit that best matches their skills and the requirements for the VR application program to be created. Bierbaum and Just summarized the dilemma that a developer faces when choosing a VR toolkit: "*Before deciding which development system is best suited to a particular application, the developer needs to know what questions to ask. This includes knowing what capabilities are required to implement the application and how to tell which systems meet those requirements.*" [68]

Literature about comparing VR toolkits is limited: there are comparisons based on feature charts [72, 73] that reveal the capabilities of different toolkits, and subjective ratings of various toolkit aspects [74, 75] that imply the level of usefulness. We have not come across literature describing more sophisticated methodology for comparing VR toolkits, which we have attempted in publication [P2].

2.4.1 3DUI Building Blocks

VR toolkits can offer more than just hardware abstraction or flexible development environments. They can provide means to build 3DUIs by including reusable components for implementing 3D interaction techniques. An early example of this is VPL's *Reality Built For Two* VR development platform that featured a node-based visual programming environment, in which data from input device nodes would flow through an editable graph of filter nodes. According to Blanchard et al., the system could be used to implement interactions such as grabbing, hit testing, and kinematics. [76]

Over the years there have been many approaches to provide similar reusable components for creating 3DUIs. In 1999 Jacob et al. [77] introduced PMIW, a description language and a visual editor for developing non-WIMP user interfaces like those of VR application programs. The visual editor of PMIW featured state diagram and data-flow graphs for defining discrete events that could enable or disable data-flows. More recently Haan and Post's StateStream approach [78] used a similar dual model with state machines and data-flow for 3D interaction techniques, which could be implemented with a Python application programming interface.

Figueroa et al. [79] presented InTml, an XML-based description language for VR application program development, which "*defines a uniform way to represent [3D interaction techniques] that is high-level, toolkit-independent, component-based, reusable, and extensible.*" InTml was further refined by Figueroa et al. [80] who introduced a visual editor, and summarized the dif-

ferent challenges in low-level implementation of 3DUIs. Fundamentally InTml language describes directed graphs with filter nodes, which Figueroa et al. dubbed "*building blocks*" that can represent a device, behavior, interaction technique, or content.

Ray and Bowman [81] pointed out that researchers have created several approaches for decomposing 3D interaction techniques into separate, reusable components. Csisinko and Kaufmann characterized such components as 3D interaction technique "*building blocks*" [27]. Typically 3D interaction techniques are modeled with state machines [82], data-flow networks [79, 83], or both [27, 77, 78], where the aforementioned components are nodes.

Valkov et al. [84] along with Ray and Bowman [81] have underlined the importance of being able to share 3D interaction techniques and their components across multiple VR toolkits without reimplementation. According to Martínez et al. that goal is hindered by the lack of agreement on a standardized set of basic interactive tasks and interaction techniques [85]. Nevertheless, several 3DUI frameworks for the purpose of sharing and reusing 3D interaction techniques across multiple VR toolkits have been proposed, for example SVIFT [86], IFFI [81], VITAL [27], and Viargo [84]. From a software architecture viewpoint, these 3DUI frameworks act as an additional software layer below the application layer, but above the VR toolkit and its subcomponents.

Wingrave's doctoral thesis [87] discussed extensively different methods and frameworks for developing 3DUIs, and provided a good overview on issues related to implementing 3D interaction techniques.

We use the expression *3DUI building block* to denote reusable components that can be modified and combined to create individual 3D interaction techniques or complete 3DUIs. As long as the 3DUI building blocks fit this description, it does not matter whether they are functions in a VR programming library such as Viargo, elements of a 3DUI description language like InTml, graphical objects in a visual programming environment, or something else.

Our expression is inspired by two sentences: firstly, Bowman et al. [5] stated that "*low-level interaction techniques and interface components are the building blocks of complete 3DUIs.*" Secondly, Wingrave and LaViola wrote in their paper that dealt with VR development challenges [7]: "*Nearly every interviewee expressed a desire for plug and play interfaces or the ability to build interfaces as easily and interchangeably as LEGO blocks.*" We employed the expression 3DUI building block first in our 3D questionnaire and in the subsequent publication [P2].

2.5 Teaching VR for Students

An abundance of literature exists about utilizing VR application programs to assist training and education, while publications about teaching VR as a subject are quite rare. Majority of such publications describe a VR course design and on occasion evaluate results from one course iteration [88–92].

There are fewer papers that consider multiple years of a VR course's history: Zara discussed lessons learned years during six years of organizing a VR

course [93]. Miyata et al. put forward student learning experience ratings averaged over several years and described student created VR application programs [94]. A paper by Häfner et al. is the only publication that has presented quantitative data for each course iteration – in this case students' ratings of different course aspects – but the different VR course iterations were not explicitly compared, statistically or otherwise [95]. According to our literature review, our publication [P3] is the first longitudinal study of a VR course, where the course evolution is studied in detail using statistical analysis to compare different course iterations.

It can be summarized that most of the VR courses described in the literature take into account the multidisciplinary nature of VR, emphasize the importance of hands-on experience, and include a group project where a VR application program is created.

Burdea estimated in 2004 that only 3% of universities and colleges around the world taught VR courses, based on his worldwide survey that found 148 universities offering courses on VR [96]. Burdea's updated VR course list from 2008 demonstrated that the number of such universities had increased to 288 [97].

Justification for teaching VR is rarely discussed in the literature: Burdea asserted that the demand for VR specialists in growing as VR technology is being applied in an increasing number of industries [96], while Stansfield mentioned the need to educate the next generation of VR researchers [90].

2.6 Main Contributions of the Thesis

The previous research discussed in this section is extended by the work at hand, which makes the following contributions: the creation of an easy-to-use open-source VR toolkit (Chapter 3) with several 3DUI building blocks, including ones that provide versatile avatar functionality (Section 3.2.1), the first comprehensive survey on 3DUI developers and their application programs (Section 4.1), the first studies with quantitative data about VR development challenges and the ranking of those challenges according to their severity (Section 4.2), novel benchmarks for comparing VR toolkits (Section 4.3), the first longitudinal study with statistical analysis on the evolution of a VR course (Section 4.4) and the lessons learned (Section 4.4.1), evidence on how VR toolkits can affect the quality of the created application programs (Section 4.4.2), and case studies involving users of VR application programs (Section 5).

3. RUIS Toolkit

RUIS is an open source VR toolkit that utilizes off-the-shelf hardware and a free to use integrated development environment. Features of RUIS have been influenced by VR toolkit requirements of Varcholik et al. [70] and the "*Reality-Based Interaction*" conceptual framework for non-traditional interfaces by Jacob et al. [98], which encompasses the themes of naïve physics, body awareness, environment awareness, and social awareness.

As mentioned in Section 2.3, the goal of RUIS is to unleash the innovation potential of hobbyist VR developers by offering an easy to use VR toolkit that can be adopted by a wide range of people. Publication [P1] introduced and explained our requirements for a VR toolkit with a *low barrier of entry*, i.e. a convenient VR toolkit that allows starting VR application program development easily and with little or no previous experience. These requirements are:

- R1. Developing applications is possible without any knowledge about compilers or linkers
- R2. Development is possible with a normal PC or laptop
- R3. Motion trackers can be simulated with a mouse and a keyboard
- R4. Input devices are abstracted and high-level data is provided
- R5. 3D selection and manipulation utilities are provided
- R6. The toolkit is free
- R7. Applications can be tested without a slow build process
- R8. Applications can be easily exported to a different computer

The RUIS toolkit fulfills the above eight requirements. Together RUIS and the aforementioned requirements are an integral part of this thesis' answer to research question 2 posed in Section 1.2. They both attempt to facilitate VR software development and address its challenges: novice VR developers can use RUIS to create VR application programs with relatively little effort, and VR toolkit creators can comply with some or all of the requirements in order to make their toolkit more convenient and usable for a larger audience of VR developers.

Requirements R1-R6 lower the barrier of starting the development of VR application programs. R7 enables a rapid, iterative development process, while R8 makes deployment and testing easier. Publication [P1] further elaborates

on how the requirements benefit teaching VR development and address specific development challenges. Publication [P1] also includes a feature chart with 13 freely available VR toolkits and how they fulfill the requirements R1-R8.

The above requirements for a VR toolkit with a low barrier of entry are based on our VR development experience and our interpretation of VR literature. Requirement R2 originates from our wish to make VR development accessible to as many people as possible. Particularly in the past some laboratories have developed VR application programs with special hardware such as Silicon Graphics workstations, which most developers do not have access to. Requirements R3 and R7 stem from our own experiences; on many occasions we have used 3D input device simulation to test our VR application program without having the actual hardware at hand, and for rapid, iterative development it is crucial that the application build process takes as little time as possible.

Requirement R1 is based on experiences shared by us and Taylor et al. [69] regarding linking and compiling VR application programs. Requirements R4, R6, and R8 were inspired by VR toolkit guidelines of Taylor et al. [69], who encouraged input device abstraction, permissive licensing, and high portability. Requirement R5 was influenced by Wingrave and LaViola [7], who reported that nearly every developer who they interviewed hoped that toolkits would have building blocks for creating 3DUIs.

A majority of our requirements are directly related to Von Hippel's [33] attributes of a high-quality toolkit for user innovation from Section 2.4. Requirements R1, R2, and R4 can be seen as manifestations of Von Hippel's third attribute about the toolkit being user friendly. Requirement R5 relates to the fourth toolkit attribute, which calls for the toolkit to contain commonly used modules that can be incorporated into custom designs. Requirement R8 is analogous to the fifth toolkit attribute, which necessitates that the innovators' custom designs can be reproduced on the manufacturer's equipment without modifications.

We have not formally validated the requirements R1-R8, as that would be very difficult. As such these requirements should be considered as general guidelines for VR toolkit creators, and not as strict necessities.

Nevertheless, some evidence exists of RUIS toolkit's capability to facilitate VR software development and address its challenges: in Publication [P1] we discovered that 45 developers using RUIS for Unity rated the lack of proper 3DUI building blocks as a significantly less severe issue than a group of 17 developers using other high-level VR toolkits. Furthermore, a statistical analysis of the other 9 issues did not find RUIS for Unity developers reporting more severe difficulties than the other group of developers. Similarly, publication [P2] demonstrated that 14 developers using RUIS for Processing rated the difficulty of learning the toolkit as significantly less severe issue when compared to 18 developers using OpenNI library.

Novice developers' VR application programs created with RUIS indicate the toolkit's suitability for hobbyist VR development purposes. A body of 28 VR application programs have been created in five different VR courses by a total

of 91 students, 71% of whom had not created VR application programs before the course. Several of the students' creations and their quality are presented in publications [P1] and [P3].

While RUIS is not currently a wildly popular toolkit, it has been adopted independently by a number of international developers who have not participated in the VR courses organized by us. RUIS online forum activity reflects the use of the toolkit, and at the moment the online forum has 58 members who we have not met before, who have created 34 topics and made 91 posts. Particularly RUIS' Kinect-controlled avatar has been popular among developers.

RUIS toolkit or components of it have been used in all the VR application programs presented in publications [P1, P3–P6] and as such RUIS forms the cornerstone of this thesis. In the following two sections we introduce the two variants of RUIS toolkit that exist.

3.1 RUIS for Processing

Development of RUIS for Processing started in 2010, when we started to add abstraction layers to our existing codebase that had been used in various VR projects. Our codebase was built on top of Processing, a simple Java-based development environment used by artists, designers, and teachers, among others [99].

RUIS for Processing is fundamentally a programming library that comes with code templates on top of which developers can create their VR application program. RUIS for Processing is not updated anymore and has become somewhat obsolete, but it can still be obtained from RUIS website [100].

Initially RUIS for Processing worked in Upponurkka, our laboratory's VR environment with two 3D stereo-projection walls [101]. We employed a custom LED tracking system to track the position of user's head and two Nintendo Wii Remote (Wiimote) controllers with color LEDs attached to them. The position tracked Wiimotes acted as 6DOF controllers, because they also provided orientation, which RUIS for Processing yaw-corrected by utilizing stationary infrared LEDs and Wiimote's infrared camera. From early on RUIS supported simultaneous use of multiple stereo 3D displays and keystone correction for projection walls (Figure 4), provided 6DOF controller simulation with a mouse and high-level functions for basic 3D selection, manipulation and navigation. This first version of RUIS for Processing was introduced in publication [P6].

In 2011 RUIS for Processing became more usable to developers outside our university, when a new version added support to PlayStation Eye camera and PlayStation Move controllers. These were used from there on as hand-held controllers and head-tracking attachments, instead of our custom LED tracking system. By spring 2012 RUIS could also utilize Kinect's full-body motion tracking, and included a simple calibration process that enabled the use of PlayStation Move controllers and Kinect-tracked avatar in the same coordinate system.

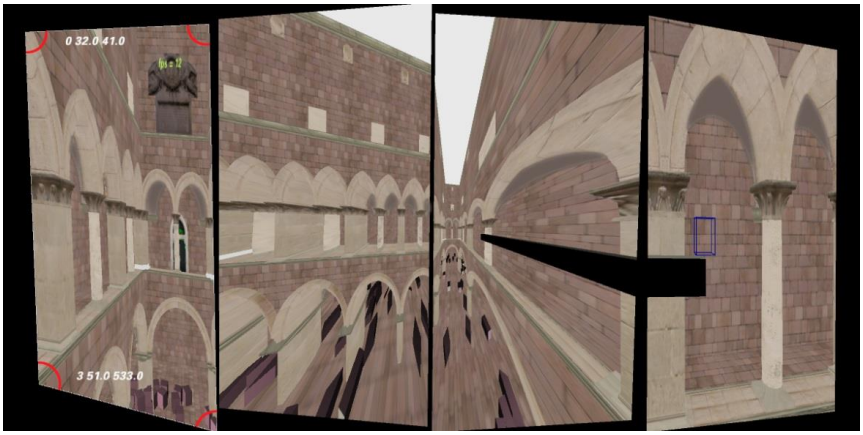


Figure 4. Projector keystone corrections and oblique perspective projection in a RUIS for Processing VR application program intended to be used in a four wall CAVE-setup.

At that point our ambitions for RUIS as a VR toolkit were too restricted by Processing's limitations for VR development, which included its slow graphics performance, poor content integration capabilities, insufficient 3D programming library, as well as its lack of vital features such as a scenegraph and built-in physics and audio engines. In an attempt to address these issues, we started porting RUIS to Unity 3D development suite, which overcomes the aforementioned limitations of Processing.

3.2 RUIS for Unity

All the features from RUIS for Processing were implemented into RUIS for Unity by spring 2013. Publication [P1] introduced RUIS for Unity, the principles behind it, the included features as they were in 2014, and a selection of VR application programs created with it. A more detailed description of RUIS for Unity is presented in the master thesis of Matveinen [102], whose work in 2012 and 2013 was instrumental in porting RUIS from Processing to Unity. The author of this thesis supervised him, designed the RUIS for Unity architecture, and worked together with him on the implementation.

In 2014 and 2015 another research assistant aided in the continued development of RUIS for Unity by adding support for Oculus Rift DK2 and Kinect 2, and extending the toolkit's 3D interaction capabilities.

At the moment RUIS for Unity supports Kinect 1, Kinect 2, Oculus Rift, HTC Vive, PlayStation Move, and Razer Hydra devices. Like RUIS for Processing, RUIS for Unity also provides an easy calibration process that can find the rigid transformations between the coordinate systems of the aforementioned devices, allowing their simultaneous use in a shared coordinate system. This enables developers to experiment and create novel VR application programs that combine multiple VR devices. For example, students of our 2014 and 2015 VR courses developed VR games that utilized Oculus Rift, Kinect, and PlayStation Move controllers in parallel.

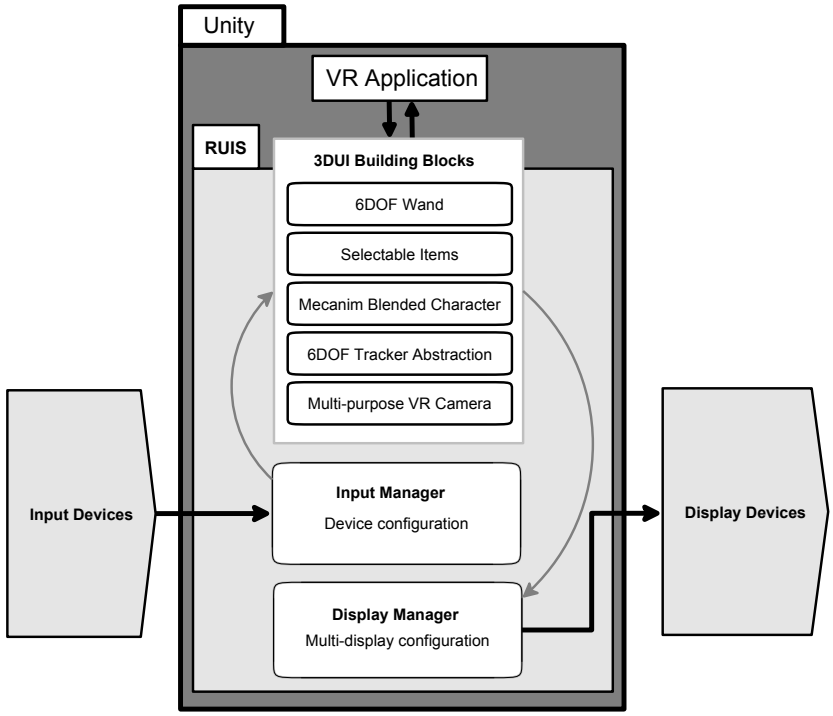


Figure 5. An overview of RUIS for Unity architecture, depicting the coupling between its modules and the devices that it interfaces with. Arrows represent data flow.

A broad overview of RUIS for Unity software architecture is shown in Figure 5: RUIS is an add-on for Unity 3D development suite, composed of various software modules. A developer uses RUIS to create a VR application program just as they would use any Unity add-on to implement ordinary software.

Roughly speaking, Unity development is carried out in the following manner: a developer uses the graphical user interface of Unity Editor to organize tree structures of game objects in Unity's scenegraph. Individual game objects act as containers for Unity components, which can be e.g. renderers, physical colliders, audio sources, or custom scripts created by the developer to define interactive behavior. A tree structure of game objects containing various components can be saved as a *prefab* into the Unity asset library. These prefabs can be summoned from the asset library as reusable elements and arranged in the Unity Editor.

RUIS for Unity comes with its own VR and 3DUI prefabs that the developer can drag and drop from the Unity asset library into their project. For instance, the RUIS input and display managers are incorporated into a single prefab, because they are both required by VR application programs that have been implemented with RUIS for Unity.

Each included RUIS prefab can be modified to suit the particular needs of a VR application program under construction. For example, the RUIS display manager can be used to configure rendering for multiple displays, where each display can have either monocular or stereo rendering and act as a normal display, a head-mounted display, or a CAVE projection wall with keystone cor-

rection and oblique perspective projection. Similarly, the RUIS input manager provides settings for individual input devices and common coordinate system settings for cases where multiple different motion tracking systems are used in conjunction.

RUIS for Unity is primarily meant to work with desktop VR devices, and its features do not function with mobile HMDs without modifications to the code.

3.2.1 3DUI Building Blocks in RUIS

When developing RUIS for Unity, we have been exploring the utilization of Unity prefabs as 3DUI building blocks. Over the years we have created and refined several high-level, reusable RUIS prefabs that can be modified and combined to create 3DUIs, and as such they fit our definition of 3DUI building blocks from Section 2.4.1.

It should be noted that the 3DUI building blocks in RUIS are not based on state machines or data-flow networks, and lack the flexibility of those approaches. Combining them is possible in Unity's scenegraph, but otherwise any meaningful interplay between the 3DUI building blocks is limited to what we have explicitly implemented. Instead, we have made them highly modifiable. Below we introduce the most prominent 3DUI building blocks in RUIS for Unity, which are summarized in Table 1.

Table 1. 3DUI building blocks included in RUIS for Unity.

Building block category	Prefab name(s)	Summary of functionality
RUIS wand	OpenVRWands	Spatial interaction tools with different VR input sources. Each wand can be configured to follow various selection and manipulation behaviors.
	PSMoveWand	
	RazerHydraWand	
	SkeletonWand *(Kinect)*	
	MouseWand	
Selectable object	Crate	Selectable object prefabs can be interacted with RUIS wands. Crate prefab is an example of a generic selectable object, while the others have selectable hinge joints.
	BigLever	
	ThrustLever	
	DoubleDoor	
Multi-purpose VR camera	RUISCamera	Supports HMD rendering, 3D stereo displays, head-tracked projection wall rendering with asymmetric frustum, and keystone correction.
Head-tracker assigner	HeadTrackers	Chooses the head tracking method at runtime based on the hardware that is available.
Character	Mannequin	Kinect-tracked avatar where each limb segment is represented by a separate, simple 3D model.
	ConstructorSkeleton	A more complex Kinect-tracked avatar with a single skinned 3D model.
	ControllableCharacter	Same as above, but with locomotion controls and HeadTrackers prefab.
	MecanimBlendedCharacter	Same as above, but the Kinect-tracked body parts can be animated separately.
6DOF motion tracker	RUISTracker	Allows combining different position and rotation tracker sources, and Kalman filtering their input.

RUIS wand prefabs abstract different 6DOF controllers and other input devices as generic spatial interaction tools with a shared interface. Currently RUIS for Unity has wand prefabs for 2D mouse, HTC Vive, Razer Hydra, PlayStation Move, and Kinect hand tracking. Figure 6 shows the components of a

PlayStation Move wand, including the RUIS Wand Selector component that is common to all RUIS wand prefabs. The developer can choose which button is used to select an object, detailed behavior of the selection button such as whether it acts as a toggle, the length of the selection ray and which object layers it affects, as well as how the position and rotation of the selected object are manipulated. In essence each RUIS wand prefab is a 3DUI building block for a 6DOF manipulator whose attributes are chosen to determine its selection and manipulation behavior. Details of the selection and manipulation process are described in publication [P1].

Figure 6. Attributes of the RUIS PlayStation Move wand prefab that can be modified via Unity Editor's graphical user interface.

RUIS offers two types of 3DUI building blocks for creating selectable items that can be interacted with by using the RUIS wands: 1) a generic *selectable object* that can be manipulated in 6DOF and affected by physical forces if needed, and 2) a *selectable hinge joint* for implementing levers, wheel controllers, doors, and hatches that can be swivelled by RUIS wands.

The most unique high-level 3DUI building blocks of RUIS are the character prefabs, especially the *Mecanim blended* character *(MBC*) prefab. The MBC integrates motion tracked real-time avatar with locomotion controls and animation blending. RUIS supports motion tracking via Kinect and can also receive data from other motion capture systems that can stream data into Unity.

The MBC prefab is relatively flexible; it includes a head-tracked first-person viewpoint camera that can render into HMDs, which allows immersive VR experiences where the user can see their avatar reflect their own body pose in the virtual world. A third-person viewpoint is also possible. Several MBC instances can be utilized concurrently in multiuser VR application programs.

We created the MBC prefab to experiment with hybrid locomotion where the avatar can be moved in three ways: by real-time motion capture, by using a hand-held controller, and by physics engine. All physical movement by the user within the range of the motion tracker will be mirrored by the MBC avatar, whether taking steps, crouching, leaning, or somersaulting.

If the user wants to travel beyond the range of the motion tracker virtually, they can utilize hand-held controllers such as the HTC Vive to make the MBC avatar run, walk, jump, or perform other actions specified by the developer. When such artificial locomotion is triggered, a predefined animation can be blended into individual avatar body parts so that the avatar appears to move naturally in the virtual world. For example, in the case of walking by pushing an analog stick of a controller, a walk-cycle animation will be temporally blended into the avatar's legs for the duration of the walk, while the rest of the avatar's body keeps mirroring the motion-tracked user. Developers can replace the MBC prefab's default rigged humanoid 3D model and animations with their own.

If so desired, the MBC avatar can take part in Unity's physics simulation, so that the user can push virtual objects with their limbs, step on objects, climb stairs, fall down into a virtual ravine, etc. An example of a VR application program employing the MBC prefab is presented in Section 5.4.

Figure 7 illustrates the most important components and their attributes in the MBC: pivot determines the location of the avatar's yaw rotation axis, ground settings are related to friction and jumping, while locomotion attributes concern traveling through the use of hand-held controllers. The skeleton controller settings determine how the motion tracking data poses the avatar, and whether the individual avatar limbs (bones to be more specific) will be scaled to match the limb sizes detected by the motion capture system.

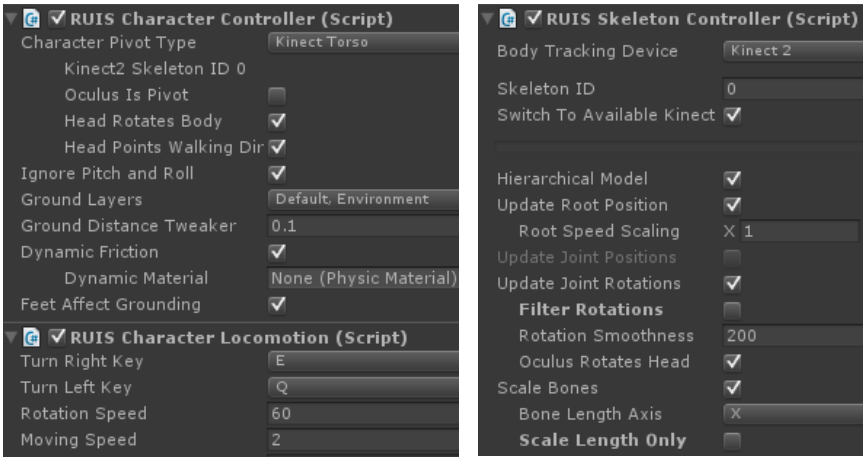


Figure 7. Various attributes of the Mecanim Blended Character prefab: (Left) Avatar controller settings and a partial list of the locomotion attributes. (Right) Settings for posing and scaling the avatar.

Other 3DUI building blocks provided by RUIS for Unity are: a *6DOF motion tracker* abstraction with optional Kalman filtering that allows separate sources for position and rotation tracking, a *multi-purpose VR camera* that works together with the RUIS display manager, and a configurable, automatic *head-tracker assigner* for those VR application programs that are intended to work with several hardware configurations. In general we have attempted to follow

the principle of making RUIS prefabs and their components to adjust themselves to the detected hardware, when possible.

Each RUIS component is a C# script, which can be edited to implement such interaction behavior that is beyond of what can be achieved by configuring the component's attributes in Unity Editor's graphical user interface. For example, the default behavior for RUIS Selectable component is to have its position and rotation manipulated upon selection by a RUIS wand. Using C# inheritance, the RUIS Selectable can be extended with another class, where the default behavior could be replaced with something else, e.g. triggering an animation upon selection.

The RUIS prefabs presented above correspond to our definition of 3DUI buildings blocks. As illustrated in Figure 6 and Figure 7 they can be modified, which affects the subsequent 3D interaction. The RUIS prefabs can also be combined to create a more complex 3DUI. For example, the selectable object and hinge joint prefabs in RUIS can be used to create virtual control panels and 3D widgets, which can be interacted via RUIS wands. Similarly, a RUIS wand or a 6DOF motion tracker abstraction can be parented under an MBC instance, so that the former are affected by the MBC locomotion and stay with the avatar regardless of its movement.

3.2.2 Unique Aspects of RUIS for Unity

As affordable consumer VR hardware has become available recently, software support and toolkits for that hardware has rapidly emerged. Both Unity and Unreal development suites have added native support for consumer HMDs, and developers can utilize consumer VR controllers by installing official plugins from Valve and Oculus VR. These additions enable a plug-and-play approach to VR development. This makes it simple to get started with creating VR application programs, and in this sense RUIS for Unity has more competition than before. Similarly, recent Unity plugins like Newton VR [103] and VRTK [104] offer VR developers a rich set of selection and manipulation 3DUI building blocks that go far further than those of RUIS, providing many more ways to implement 3DUIs for VR application programs. It is therefore reasonable to ask how relevant RUIS is anymore in the year 2016.

Fortunately, RUIS for Unity still contains a few unique features that should make the toolkit appealing to some developers: firstly, there is the aforementioned ability to easily pair the coordinate systems of multiple motion trackers through a semi-automated process, which allows the utilization of different VR input devices in a shared coordinate system. For example, developers can create VR application programs where a Kinect sensor is used in conjunction with a HMD, so that a Kinect-tracked avatar body replaces the user's body when viewed through the HMD.

Secondly, the motion-tracked avatar functionality provided by the MBC prefab of RUIS is unmatched. As discussed in the previous section, among other things the MBC offers a simple way to interactively blend character animation sequences and real-time motion capture data on the level of individual limbs, an option for using traditional locomotion controls as part of the MBC's hybrid

locomotion scheme, and a possibility to have the avatar affected by physics simulation.

Lastly, RUIS supports the use of multiple stereo 3D displays and keystone correction for projection walls such as those employed in CAVEs. While this is not a unique feature, the implementation in RUIS fares very well against other modern Unity-based VR toolkits with similar functionality, as shown in a recent comparison by Ritter et al. [75].

4. Virtual Reality Software Developers

The center of interest for this thesis lies in VR developers and their challenges, which are examined in this chapter. Methodology for comparing VR development challenges and VR toolkits are also presented, along with a VR course where we taught VR development for students. Most of the results introduced in this chapter are based on data that we gathered with a 3DUI questionnaire, which over the years has been answered by a total of 167 developers.

4.1 3D User Interface Survey

In 2011 we formulated an online questionnaire intended for developers who have developed at least one VR, AR, or MR application program with a 3DUI that employed spatial input devices such as 6DOF controllers or motion capture suits. The questionnaire gathered information about the background of the developer, the 3DUI application program that they had developed, the hardware and software used, challenges experienced during the development, and their thoughts on an ideal 3DUI toolkit. The questionnaire and its description of participant requirements can be viewed online [105].

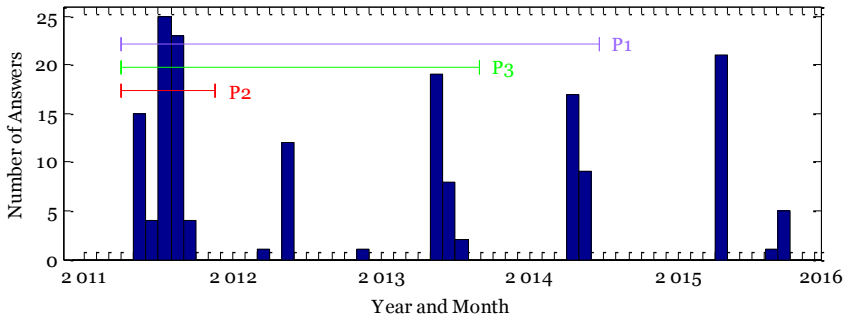


Figure 8. 3DUI questionnaire answer histogram. The horizontal line segments indicate which answers were used for publications [P1–P3].

The distribution of answers collected with the 3DUI questionnaire over time is presented in Figure 8. Each year, in the beginning of summer, there was a spike in the number of answers when students of our VR course were asked to fill the questionnaire. All in all the questionnaire has been answered by 76 developers who have not been our students. A vast majority of these answers, 57,

are from 2011, when the existence of the 3DUI questionnaire was vigorously made known to 3DUI practitioners in order to gather data for publication [P2].

From those questionnaire participants who have not been our students, only 17 answered after the release of Oculus Rift DK1 in 2013. Therefore our 3DUI questionnaire data regarding non-student developers mostly represents the 3DUI development era just prior to Oculus Rift.

Publication [P2] examined the first batch of data collected via the questionnaire, which included answers from a total of 71 participants: 14 students of our VR course, 13 colleagues, and 44 participants that we were not familiar with before. The participants were divided into three groups: student, hobbyist, and professional developers, and their demographics were presented. A majority of the participants were 20-29 years old professionals, had three years or less of 3DUI development experience, and had created at most three 3DUI application programs. The full details of the demographics, such as the developers' country of residence (Figure 9), are presented in publication [P2].

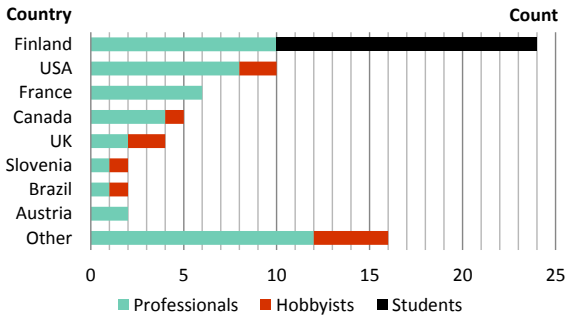


Figure 9. Countries where publication [P2]'s surveyed developers were based in.

Statistics about 56 unique 3DUI application programs were also introduced, which excluded our students' 3DUI application programs that were all made with RUIS for Processing, because that would have introduced a bias into the results. Exactly half of the reported application programs were created for VR use, while the rest were AR and MR application programs. They were categorized into research, hobby, and commercial projects. The majority of the 3DUI application programs were implemented between 2009 and 2011 for research purposes, by up to 4 developers, utilizing C/C++ programming language, and used by less than 50 people. The popularity of different input and output devices utilized by the 3DUI application programs were also presented, as well as the toolkits and programming libraries used in their development. Detailed information about the surveyed 3DUI application programs, such as the application domain distribution (Figure 10), is included in publication [P2].

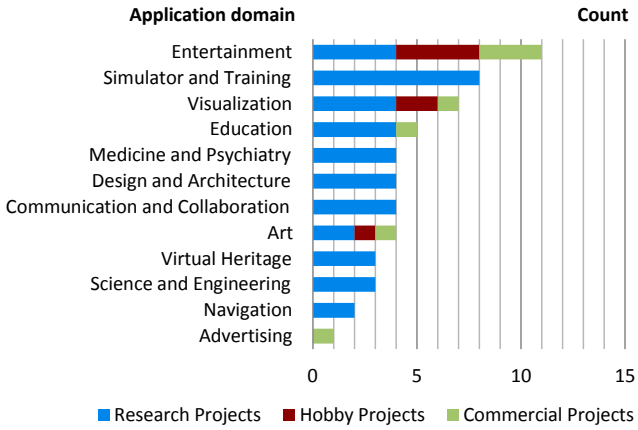


Figure 10. The most common application domains among the 56 different 3DUI application programs from the survey of publication [P2].

Furthermore, the survey [P2] showed the occurrence of different 3DUI features in the application programs and whether they were inherited from a software toolkit or implemented by the developers themselves. The two most common features were navigation and object manipulation techniques, which were present in 75% and 70% of the 3DUI application programs respectively.

Surprisingly, our survey demonstrated that inheriting 3DUI features from toolkits was rare, especially with complex features. For example, only 12% of the 3DUI application programs that included navigation techniques and 10% that included object manipulation techniques inherited their respective features. Overall it was more common for developers to implement a feature than to inherit it from a toolkit, which implies a low rate of feature reuse in the 3DUI developer community. This is in line with Wingrave and LaViola's observation that "*it remains easier to build than to reuse*" [7].

Published in the first quarter of 2012, publication [P2] further elaborated on our beliefs regarding hobbyist innovation that were first expressed in publication [P6] from 2011, and made recommendations on how companies could further encourage such innovation: "*A positive feedback loop could emerge between 3DUI hobbyist community and related technology companies, where innovations spread and accumulate between commercial developers and hobbyists [...] Any device manufacturer, that wants to tap into the stream of hobbyist innovation, should aim to remove unnecessary barriers between their device and its potential developers.*" As far as we know [P2] is also the first publication to tie hobbyist innovation in the field of VR to the existing user-led innovation literature: we refer to Hippel and Katz [65] as well as Aoyama and Izushi [38].

4.2 Development Challenges

Publication [P2] introduced how existing survey methodology could be applied in a novel way for quantitative measurement and comparison of challenges in VR and 3DUI development: we devised 18 statements regarding different de-

velopment difficulties that can be rated on a Likert scale according to their severity. A collection of such ratings – possibly coupled with other data – can then be used for statistical analysis of challenges in VR development. Likert ratings and statistical analysis are widely employed in VR and 3DUI user studies, but to the best of our knowledge publication [P2] was the first to utilize them for examining 3DUI development difficulties.

Several of our 18 difficulty statements were inspired by Wingrave and LaViola's list of development issues [7], while the rest stemmed from our own experience. For example, getting started with VR development was quite challenging for the author of this thesis in 2006 when he first begun working with Upponurkka virtual environment [101] and the custom VR software that was used prior to RUIS: setting up a development environment with the required programming libraries and linking them correctly, interfacing with exotic VR device drivers, having access only to low-level input data from those devices, etc.

This led the author of this thesis to wonder how big of an obstacle such initial difficulties would be for aspiring VR hobbyists, and whether that could lead them to abandon their task of developing VR applications altogether. In an effort to shed light on the matter, the development difficulty statements were formulated so that 10 of them concerned early difficulties that could be experienced at the beginning of the development process, while 8 were about later difficulties that could occur subsequently.

In our 3DUI questionnaire the statements were rated on a seven point Likert scale, where 1 indicated strong disagreement and 7 indicated strong agreement. The participants were asked to base their ratings on the development experience of the particular VR application program that they had described earlier in the questionnaire.

All the statements were constructed in a manner where a higher rating signifies a more severe difficulty. That way the rating process is consistent for the questionnaire participants and they can easily indicate whether some difficulties are more severe than others. This also allows us to compare the severity of the difficulties.

Below we present the 10 statements regarding difficulties that can take place early in VR and 3DUI development:

A1. The input devices required by my 3DUI application were too expensive

A2. The output devices required by my 3DUI application were too expensive

A3. Getting input device drivers to work was difficult

A4. There were too many steps required between connecting the input device for the first time and successfully streaming data from the device into my application

A5. Device input data was too low-level for quickly getting started with my 3DUI application

A6. Lack of documentation or tutorials about the 3DUI toolkit made the development difficult

A7. The 3DUI toolkit had a steep learning curve

A8. The development was difficult because the 3DUI toolkit had a bad programming interface

A9. Programming in general was difficult

A10. Creating mathematical algorithms required by my application's 3DUI was difficult

It should be noted that in this section and in publication [P2] the statements refer to "*3DUI application [program]*" and "*3DUI toolkit*". However, within these two expressions the abbreviation "3DUI" is interchangeable with "VR", at least in the context of this thesis.

The following 8 statements concern difficulties that can occur later in the development process:

B1. Input device performance was poor

B2. There were bugs in the 3DUI toolkit that I used for developing my 3DUI application, making the development difficult

B3. Lack of proper 3DUI building blocks made it difficult to develop my 3DUI application

B4. Each added 3D interaction feature increased application complexity, making the development difficult

B5. Constant testing and re-implementation was required, making the development difficult

B6. Testing of the application's 3DUI could not be carried out properly with just mouse and keyboard, making the development difficult

B7. Teamwork was difficult

B8. Legal status of using unofficial drivers and libraries for commercial purposes was unclear

Publications [P1–P3] examined separate subsets of developers who had answered our 3DUI questionnaire, juxtaposing difficulty statement ratings from different groups of VR and 3DUI developers. For example in publication [P2] we compared 30 inexperienced and 27 experienced 3DUI developers, and discovered that the inexperienced group rated A8 and A9 as significantly more severe.

In the remainder of this section we rank the VR development difficulty statements by their severity, using difficulty ratings from all the developers who described creating a VR application program in their answers to our 3DUI questionnaire between 2011 and 2016 (Figure 8). Those 35 participants who

reported developing AR and MR application programs were not included, because we specifically wanted to find the most severe VR development challenges, in accordance with research question 1 of this thesis.

Consequently we have difficulty statement ratings from a total of 132 VR developers, 91 of whom were students in our VR course between 2011 and 2015, and 41 other developers who have answered our 3DUI questionnaire. The latter group is generally more experienced, containing 28 professionals, 9 hobbyists, and 4 students. Therefore the total pool of 132 participants consists mostly of novice VR developers, whose demographics match very closely to the developer demographics presented in our survey [P2].

Our students used 6DOF controllers and Upponurkka virtual environment [101], latter of which was replaced with Oculus Rift DK2 in the 2015 course. Utilization of different input and output devices by the other 41 VR developers was the following: 11 Kinects, 11 Wiimotes, 11 cameras, 10 6DOF controllers, 6 data gloves, 21 monitors, 15 projectors, 7 CAVE or surround-screen displays, 5 Oculus Rift HMDs, and 4 unspecified HMDs.

The difficulty statement ratings between our students and the other 41 VR developers were quite similar overall. Only one statistically significant difference was found, using a Wilcoxon rank-sum test with Bonferroni correction ($z = 3.4$, $p < 0.001$, $r = 0.29$): our students rated statement B6 as more severe, indicating that the group with more professionals had a better access to VR hardware, which is not surprising.

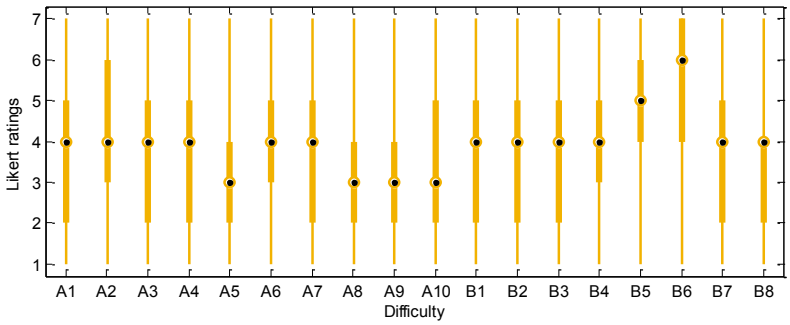


Figure 11. Severity of VR development difficulties (boxplot of Likert ratings).

Finally we examined whether statistically significant differences existed between the 18 statements, as rated by the 132 VR developers. The Likert rating distributions for each statement are approximated by the boxplot of Figure 11. A significant effect of difficulty statements was detected with a Friedman test ($\chi^2(17) = 267.4$, $p < 0.001$). Post-hoc multiple comparisons with Bonferroni correction was performed using MATLAB, revealing significant differences between the statements. Table 2 list those differences as well as the associated effect sizes that were calculated with pairwise comparisons utilizing a Wilcoxon signed-rank test. An effect size is a measure of observed effect magnitude, which in our case is the magnitude of difference in severity. From statistics literature we obtained the formula ($r = Z/\sqrt{N}$) to calculate effect sizes for our

test, and thresholds of 0.1, 0.3, and 0.5 for small, medium, and large effect sizes, respectively [106].

Statistically significant differences and the relative severity between all the 18 statements are illustrated by Figure 12, which was created with MATLAB's *friedman* and *multcompare* functions using all the VR developers' ratings. The more severe ratings a statement has received, the farther right it appears on the chart. The comparable severity ranking between those statements whose comparison intervals overlap in Figure 12 should be considered only as weakly indicative. It is only when comparison intervals between two statements are disjoint, there is sufficient statistical evidence to say that the rightmost of the two difficulties is significantly more severe.

Table 2. Pairwise comparisons: a significant difference exists for each table cell with a number (effect size), so that the (row) statement on the left is more severe than the (column) statement above. Medium and large effect sizes are denoted with light and dark gray color, respectively.

	More Severe Than																
Statement	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B1	B2	B3	B4	B5	B6	B7	B8
A1					0.28				0.27									
A2					0.36			0.30	0.36	0.22		0.23						0.26
A3					0.22													
A4					0.32				0.28									
A5																		
A6					0.37			0.34	0.37	0.24								0.26
A7					0.29				0.30									
A8																		
A9																		
A10																		
B1					0.21													
B2																		
B3					0.26				0.27									
B4					0.32				0.31									
B5	0.26		0.36	0.28	0.47		0.35	0.42	**0.50**	0.39	0.37	0.37	0.33	0.37			0.26	0.38
B6	0.35	0.26	0.40	0.35	0.46	0.30	0.39	0.44	0.49	0.42	0.38	0.41	0.38	0.37			0.34	0.41
B7					0.27				0.26									
B8																		

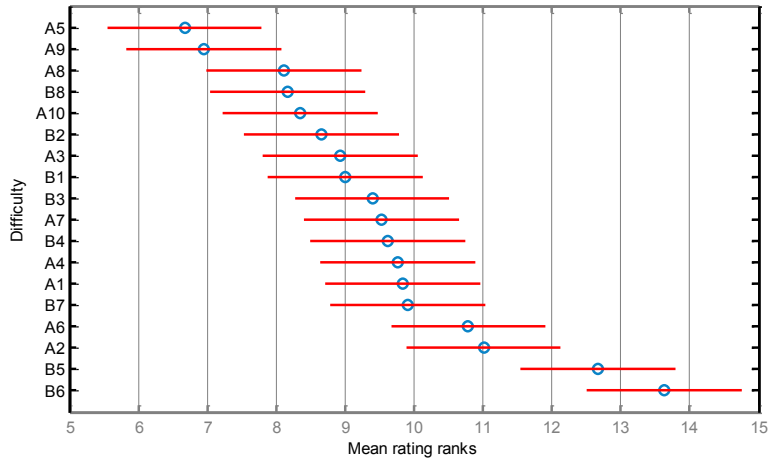


Figure 12. Mean ranks (circles) computed by Friedman test for ratings of A1-A10 and B1-B8. Ratings between two statements are significantly different if their comparison intervals (horizontal line segments) do not overlap. Rating severity increases from left to right.

As can be seen from Figure 12 and Table 2, both statements B5 and B6 are rated significantly more severe than the rest of the statements, with the exception of non-significance between B5, A2, and A6. In the same vein, Figure 12 and Table 2 demonstrate that A2 and A6 are the most severe of the early difficulties in VR development. On the other end of the spectrum, A5 and A9 are the least severe difficulties.

There are 12 difficulty statements in the middle of Figure 12, which are not significantly different from each other in terms of severity. Therefore any severity differences between them, as suggested by their horizontal location, are indicative at best.

These results directly address research question 1, by revealing what the most severe challenges in VR software development are according to the 132 VR developers who answered our 3DUI questionnaire. This analysis of the consolidated difficulty ratings data is previously unpublished, while different subsets of the same data have been utilized before in publications [P1–P3].

4.2.1 Difficulty Concepts Gathered with Open-Ended Questions

The statements A1-A10 and B1-B8 were designed to cover the most common 3DUI development difficulties. However, a finite, predefined list of statements is not likely to contain all the relevant challenges of developers who work with 3DUIs.

Therefore our 3DUI questionnaire included two open-ended questions dealing with 3DUI difficulties: one question asking to describe difficulties encountered early in the development process and another question about later difficulties. When writing publication [P3], we examined the answers to these two questions, collected from 45 students who took part in our VR course between 2011 and 2013, and 69 other questionnaire participants.

We pooled the two answer sets regarding early and later difficulties together, since there were several overlapping difficulty descriptions. We extracted key ideas from the answers, compressing them into a set of 33 difficulty concepts that were perceived by the questionnaire participants.

Table 3. Number of mentions about various development difficulties by our students and other participants who answered open-ended questions of our 3DUI questionnaire.

Development Difficulty	Mentions by Students	Mentions by Others	Total Mentions
Limited VR setup access	9	5	14
Bad documentation	6	4	10
Poor tracking	7	1	8
Version control	8	0	8
Driver or library issues	4	2	6
Learning the toolkit	6	0	6
Lack of 3DUI experience	3	1	4
Poor audio capabilities	4	0	4
Time management	4	0	4
Prior knowledge required	1	3	4
Gesture recognition	0	3	3

We focused on a subset of 11 concepts that each had at least 3 mentions, which represented 72% of the overall coded answers (Table 3). Most difficulties listed in Table 3 were reported by students of our course, because answering the

open-ended questions was mandatory to them, whereas the other questionnaire participants could skip those questions.

Majority of the described difficulty concepts of Table 3 were covered by our predefined difficulty statements, including the most often mentioned issue of "Limited VR setup access", which is directly related to statement B6 that was found to be the most severe difficulty in Section 4.2. The following concepts were not covered by our difficulty statements: issues with version control that were due to our students using Unity, lack of 3DUI experience, poor audio capabilities of the utilized VR toolkit, time management of students who had trouble allocating enough time for our course, the amount of prior knowledge that was required to utilize the VR system, and issues with gesture recognition. Looking at these specific concepts and the number of mentions they received, we argue that only the latter two and the lack of 3DUI experience could be considered as premises for new predefined development difficulty statements. These results were intended to be included in publication [P3] as additional material, but they were finally dropped in order to shorten the already lengthy paper.

4.3 VR Toolkit Comparisons

Publication [P2] introduced two new methods for VR and 3DUI toolkit benchmarking: feature-based and difficulty-based benchmarking, both of which are based on the idea of utilizing data gathered from developers using a questionnaire such as ours. In both schemes the questionnaire answers are partitioned by the toolkits used in the development of the reported VR application programs, and each toolkit partition gets ranked by examining development experiences or artifacts particular to the benchmark in question.

The feature-based benchmark ranks VR toolkits by the proportion of inherited and implemented 3DUI features among the surveyed VR application programs. The idea of the benchmark is that developers, who want to avoid implementing certain features by themselves, could easily determine the toolkits that are most often utilized for inheriting the desired features. As such the feature-based benchmark can facilitate reuse of VR and 3DUI software components.

The following list of 3DUI features was presented in publication [P2]:

F1. 3D stereographics
F2. Head tracked view rendering
F3. Full-body interaction
F4. Two-handed interaction
F5. Finger interaction
F6. Gesture recognition
F7. Navigation techniques
F8. Object manipulation techniques
F9. Physics engine
F10. 3D audio

The above list was not intended to be all-encompassing, and other questionnaires could include different features. An example of using the feature-based benchmark was demonstrated in publication [P2]: from a group of 4 different programming libraries and toolkits to be ranked, Microsoft Kinect SDK had the highest inheritance ratio of the full-body interaction feature among all the 3DUI application programs that employed it.

The difficulty-based benchmark ranks VR toolkits according the ratings of the development difficulty statements. The underlying principle is that the toolkits with low difficulty ratings are easy to use. Thus the difficulty-based benchmark can assist developers, particularly novices, in selecting the least difficult VR toolkit for their development needs.

Publication [P2] suggested that difficulty-based benchmarks should mostly focus on ratings of statements A3-A8 and B2-B6, because the other difficulty statements are not directly related to 3DUIs or VR toolkits.

As described in Section 3, we used the difficulty-based benchmark in publications [P1] and [P2] for comparing other toolkits to RUIS for Unity and RUIS for Processing, respectively. In publication [P3] we used the-difficulty based benchmark to track the evolution of the RUIS toolkit, by comparing RUIS for Unity and two versions of RUIS for Processing. Publication [P3] also explored further ways of benchmarking toolkits, by juxtaposing the number of 3D models, sound assets, and complex features in VR application programs that were created with the three versions of RUIS toolkit. Different toolkit users' willingness to showcase their VR application program to potential employers was also utilized as a benchmark.

All in all our 3DUI questionnaire has quite an extensive set of questions, which take time and energy to answer. If the purpose of a questionnaire is to merely produce data for toolkit benchmarking, then it is sufficient to use a very narrow subset of items from our 3DUI questionnaire.

4.4 Virtual Reality Course for Students

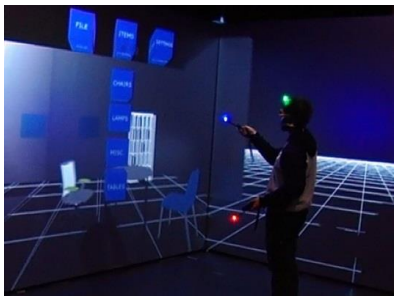
From 2011 to 2015 we organized an annual VR course in Aalto University, which each year was completed by between 12 to 28 students. Publication [P3] introduced the VR course and detailed its structure, learning goals, student deliverables, assessment, and instructor involvement.

The course focused on teaching VR application program development for students. It was organized as a project course, where students formed groups to develop VR application programs together. We selected this approach because project courses suit the interdisciplinary nature of VR development well, which requires understanding programming, interface design, human psychology, and content creation.

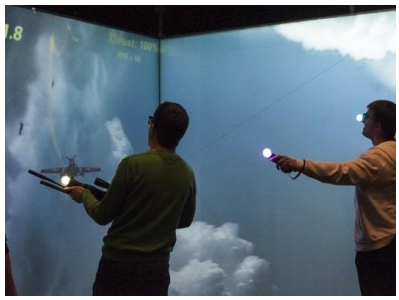
In publication [P3] we emphasized the importance of hands-on experience in VR courses and how organizing a VR course with such a hands-on approach is more feasible now than ever before, due to the availability of affordable consumer VR products. We pointed out that nowadays courses involving VR de-

velopment do not need to be complicated for students because of development tools that are easy to setup and work with popular, interpreted programming languages.

Learning assessment results for our course were featured in publication [P3]. The assessment was conducted in 2015 with an additional question form including open questions and Likert statements, which the students used for self-evaluation. Statistical analysis of the statement ratings indicated that the students learned about basic VR concepts, 3DUIs, VR technology, and VR application program implementation. The most evident result was that the students expressed becoming better at contributing to VR application program creation. Answers to open questions suggested that overall the students met their learning goals during the course.



(a) Room Planner, 2011

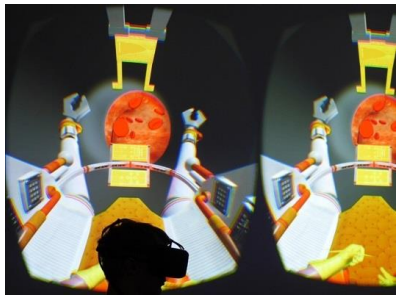


(b) Air Supremacy, 2012

(c) Battletruck, 2013

(d) Wheelchair Champion, 2014

(e) Curling Simulation, 2014



(f) Antibody, 2015

Figure 13. A selection of VR application programs created by the students of our VR course between years 2011-2015.

Publication [P3] also presented the small adjustments that we made to the VR course within the five years that it was organized. Through those years RUIS toolkit was utilized by all but one student group. More detailed analysis was performed on the first three course iterations, during which the underlying platform of RUIS toolkit was switched from Processing to Unity. The effect of our improvements to RUIS and the course was investigated by comparing the student-created VR application programs, student feedback, and their answers to our 3DUI questionnaire.

Our quantitative and qualitative analysis indicated that the latest of the compared course iterations was the most successful with regard to the VR application program quality and student feedback. Examples of students' work are presented in Figure 13 and in a video that is available online.[1]

4.4.1 Practical Considerations for VR Course Organizers

In publication [P3] we outlined the lessons learned during the five years of arranging the VR course by presenting multiple practical considerations for VR course organizers. These considerations are summarized below.

Besides having practical and theoretical VR knowledge, VR course instructors should be familiar with human-computer interaction and immersive technologies. In order to avoid overwhelming the VR course students with the amount of knowledge that is involved in developing a VR application program, there should be some prerequisites for the students. For example, it could be required that they have already completed courses in related subjects such as computer graphics or user interface design.

The VR course students should have an ample access to VR hardware, so that they can readily develop and test their VR application programs. We recommend that the teaching laboratory has one VR workstation per student group, and that the students can also borrow VR peripherals. Teaching laboratory equipment and lecture material should be kept up to date if possible, as the field of VR is rapidly advancing.

Although HMDs are finally affordable, CAVEs still have their place in VR education, because they make it possible for an entire class of students to see VR application programs being demonstrated clearly. Furthermore, relying solely on HMDs discourages the development of multiuser VR application programs, as each HMD requires its own computer, which necessitates implementing a VR application program running on a network of computers.

On some VR courses instructors might want to focus on teaching VR application program development utilizing a specific VR toolkit. In those cases the choice of the VR toolkit should be based on the course's learning goals, teaching methods, available hardware, and the background of the students. Moreover, if the chosen VR toolkit does not require experience with compilers and linkers, then the course staff needs to spend less time on helping the students with technical issues such as setting up the development environment.

[1] Student-created VR application programs from 2013 and 2014: https://youtu.be/09MPIJmGTU8

Other factors that should be taken into account when choosing a VR toolkit include the quality of the toolkit documentation and the size of the developer community. Both are significant resources for the students who will use the VR toolkit. If the visual appearance of VR application programs is important for the VR course, then the chosen toolkit should contain a 3D scene manager and an asset pipeline.

4.4.2 Effect of VR Toolkits on VR Development

Our students utilized the first version of RUIS for Processing in the 2011 VR course, an updated version with PlayStation Move support in 2012, and the newly created RUIS for Unity in 2013. Because RUIS received extensive updates between 2011 and 2013, publication [P3] focused on those first three course iterations, which were subjected to a detailed comparative analysis. We examined the differences in the student-created VR application programs and the students' development experiences, finding statistically significant differences in several metrics: students from the 2013 course experienced less severe development difficulties with RUIS for Unity when compared to RUIS for Processing of earlier courses. The use of RUIS for Unity toolkit led to increased utilization of graphics and audio assets, as well as increased complexity in the VR application programs. Students from 2013 who used RUIS for Unity were also more willing to showcase their VR application programs for prospective employers.

It is not surprising that the quality of a VR application program is influenced by the VR toolkits utilized in its development. With the aforementioned quantifiable distinctions, publication [P3] provided evidence about the possible ways in which the choice of a VR toolkit can affect the development experience and the resulting VR application program.

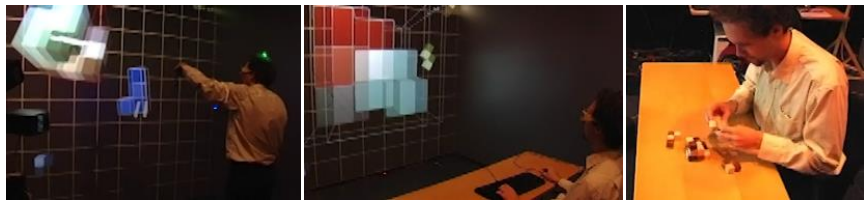
Our analysis of different groupings of students in publication [P3] acts as an example how different software development aspects can be isolated and studied: software project courses provide a controlled setting where factors like the utilized software toolkit can be examined, if the course organizers are willing to vary them between students or course iterations. Naturally this requires collecting data from the students and their projects, necessitates certain consistency in the skills and ambitions between the students, and does not allow extensive changes in the course execution if the study spans several course iterations. Such research could possibly reveal other interesting effects that can result from the choice of a software toolkit.

5. Case Studies on Virtual Reality Users

While the previous chapters have concentrated on VR developers and matters related to them, this chapter examines the users of VR application programs. We describe four VR application programs, each of which was created by us, utilizing RUIS or its components. The related case studies will also be presented, revealing the experiences of the VR application program users. The analysis in those studies contributes shards of knowledge regarding research question 3, about what benefits immersive VR offers to software users in contrast to traditional application programs. It should be noted that answering research questions 1 and 2 is the main focus of this thesis, and the research question 3 is a secondary interest, worthy of further exploration.

5.1 3DUI Puzzle

Publication [P6] provided some justification for utilizing VR and 3DUIs in specific use cases instead of traditional interfaces, but with possible caveats related to fatigue. We presented a user study with 16 participants, of whom 11 were novices and 5 experts in 3DUIs. Their task was to solve a 3D puzzle within a 10 minute time limit in three different settings (Figure 14): employing a two-handed VR interface created with RUIS for Processing, a traditional mouse and keyboard interface, and a real world puzzle. The publication was submitted to the IEEE 3DUI Contest 2011, where the VR interface placed 4th out of 8 participants. The contest restricted the publication size to 2 pages and also necessitated creating a video of the results.[2]



(a) (b) (c)

Figure 14. Study participant solving a 3D puzzle with (a) VR Interface, (b) mouse and keyboard interface, and (c) real puzzle pieces.

[2] 3DUI Contest 2011 RUIS Puzzle entry video: https://youtu.be/0DJhH1MLp1k

The participants used a Likert scale to rate the VR interface significantly more intuitive and realistic than the mouse interface. When ratings regarding all three interfaces were compared statistically, the VR interface came out as significantly more fun than the mouse interface, but it was also considered to cause significantly more fatigue than the real puzzle pieces.

Unsurprisingly, 75% of the participants indicated that the real puzzle had the best performance when asked to explicitly rank the interfaces. Objective performance metrics gave only indicative results: the puzzle was completed on the first attempt by 6 participants with the VR interface, 11 participants with the mouse interface, and 15 participants with the real puzzle. This could be partially explained by learning effects as the puzzle remained the same with each interface, and every participant first used the VR interface and then the mouse interface, because we wanted to focus on the performance of the former without any accustomization effects. The five expert participants of the study, who each received two extra attempts with the VR interface, did not solve the puzzle consistently enough to draw clear conclusions about the VR interface's learning effects.

5.2 Tennis Game with Volumetric Shadows as Depth Cues

In publication [P4] we presented a novel concept of using volumetric shadows and light shafts (Figure 15) to enhance 3D spatial perception in tasks that employ motion tracked controllers. When writing the paper in summer 2012 we were not aware of earlier use of volumetric shadows as depth cues, despite extensive online search. It is only when compiling this thesis, we came across the work of Khlebnikov et al. [107], whose publication from December 2011 dealt with volumetric lighting in tumor accessibility planning. We discovered their paper through a search that included "crepuscular rays", a term which we were not familiar with in 2012. Thus, Khlebnikov et al. invented independently the concept of using volumetric lighting as a depth cue slightly before us. However, unlike in publication [P4], their lighting calculation was done in an offline process, there were no interactive volumetric shadows, and they did not perform user studies with interactive spatial tasks.

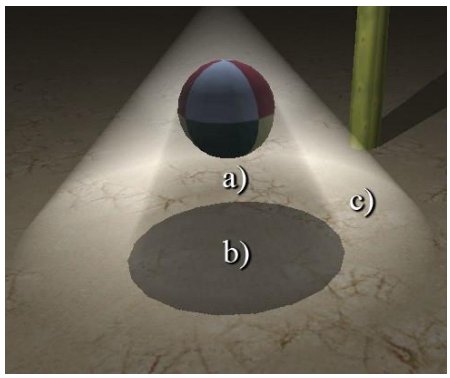
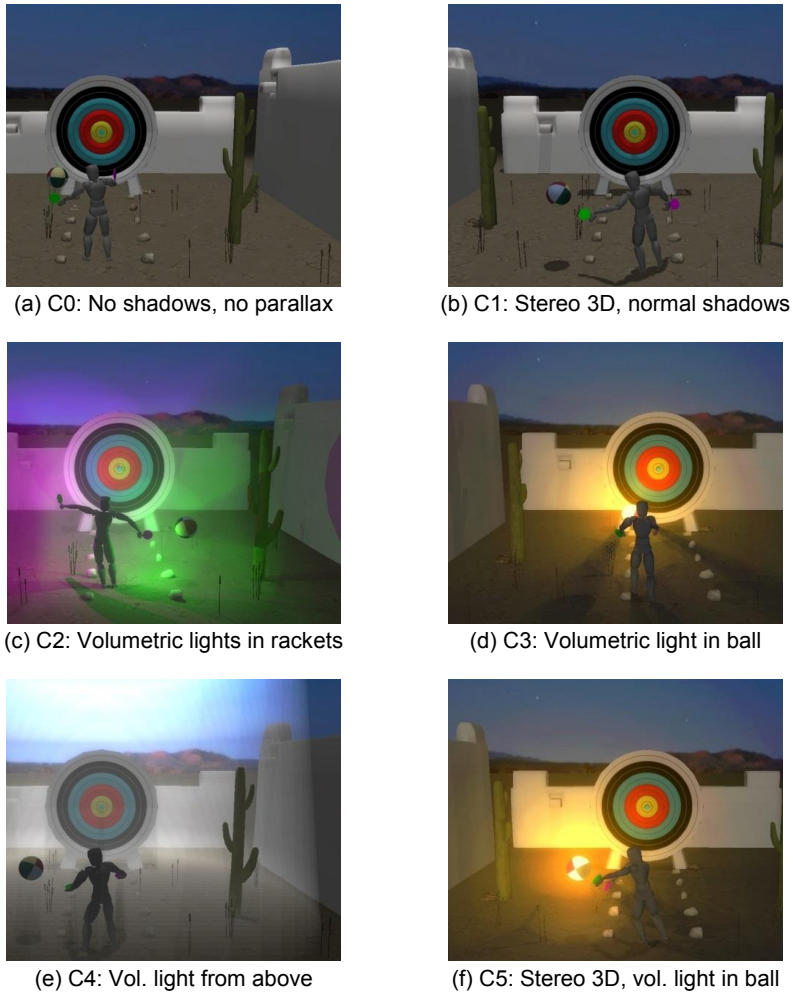


Figure 15. (a) Volumetric shadow, (b) normal shadow, and (c) volumetric light shaft.

We implemented the volumetric lighting-based depth cues into a single-player game that had some resemblance to wall tennis; the goal of the game was to use virtual rackets represented by two PlayStation Move controllers to strike virtual balls so that they would hit as close as possible to a center of an archery target. The virtual balls appeared both in stationary positions within the player's reach and moving linearly across the playing area.

The game was displayed on a 55" 3D TV, and it employed Kinect full-body tracking to give the player a better awareness of the virtual space and their avatar in it. RUIS provided the means to calibrate and map the coordinate systems between PlayStation Move and Kinect devices.



(a) C0: No shadows, no parallax

(b) C1: Stereo 3D, normal shadows

(c) C2: Volumetric lights in rackets

(d) C3: Volumetric light in ball

(e) C4: Vol. light from above

(f) C5: Stereo 3D, vol. light in ball

Figure 16. Our game's six different conditions C0-C5 and their depth cues.

Our game contained six different conditions that are illustrated in Figure 16. Each of the conditions featured a unique combination of depth cues: Condition 0 functioned as a baseline, provided only perspective depth cues, and was the only condition without head-tracked motion parallax and normal shadows.

Condition 1 exemplified “industry standard” of VR graphics with stereo 3D and normal shadows. Condition 2 utilized each racket as a volumetric light source, whereas in Condition 3 each of the virtual balls was a volumetric light source. Condition 4 featured a volumetric light from far above, and Condition 5 had stereo 3D while otherwise being the same as Condition 3.

The volumetric lighting and shadows of conditions 2-5 were our contribution to natural depth cues. These cues can be used to enhance spatial perception and to assist reaching, interception, and aiming tasks. Their effectiveness was explored in a user study, where 35 participants played through all the six conditions of our game. We exercised great care in planning the user study, which utilized a randomized within-subject design with partial counterbalancing.

Spatial task performance was examined by measuring aiming accuracy, interception rate of moving balls, and acquisition time for stationary balls. After completing each condition the participants’ user experience was examined extensively via Likert statement ratings, short interviews, and self-assessment manikin (SAM) ratings for pleasure, arousal, and dominance levels [108].

In the analysis phase we included only those 30 participants who we had tested to have a stereoscopic acuity better than or equal to 120 arc seconds. Participant interviews contained descriptions that were condensed into 17 dichotomous variables, which were explored via correspondence analysis and by correlating them with the Likert and SAM ratings.

The results of our analysis suggested that volumetric lighting can impact spatial task performance and user experience positively or negatively, depending on the particular lighting configuration. The high-above volumetric light source of Condition 4 turned out to be the most promising of the volumetric light setups; even though it featured monocular rendering, there were no statistically significant differences in spatial task performance metrics when compared to Condition 1 that was the best stereo 3D setup and represented a typical VR configuration. Conversely, Condition 4 had a significantly higher moving ball interception rate when compared to two other conditions while Condition 1 did not.

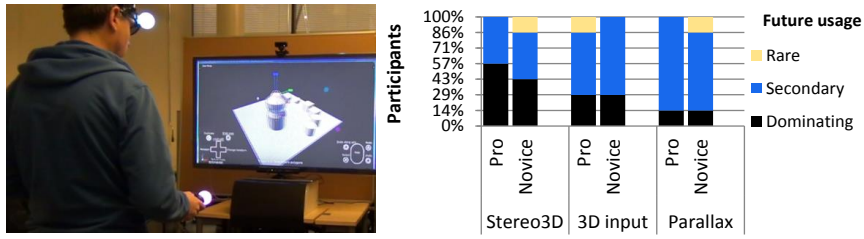
Our analysis revealed that overall the most satisfying user experience and the highest level of mastery was provided by Condition 1 with its stereo 3D and normal shadows. Nearly half of the study participants reported improved spatial perception in conditions that had volumetric lighting. However, the study participants were divided in their opinion about our game’s use of volumetric light sources: some participants were distracted by them while others were pleased with the interesting and exciting lighting.

At the end of publication [P4] we summarized our results as a list of lighting guidelines for improving spatial perception. Further user studies need to be conducted to quantify more precisely how well volumetric shadows can improve spatial perception in monocular and stereo 3D conditions, as opposed to normal shadows.

5.3 3DUI for Blender

Publication [P5] explored the benefits that immersive technology could offer to 3D artists. We introduced a custom 3DUI for the popular Blender 3D modeling software [109], which is available as open source. The 3DUI employed RUIS for Processing to handle device input, a 55" 3D TV for stereo 3D, and PlayStation Move controllers for 3D input and head tracking. Like the 3DUI puzzle (Section 5.1) from two years earlier, the 3DUI for Blender was submitted to IEEE 3DUI Contest 2013. There it was rewarded with the "Best Low-Cost, Minimally-Intrusive Solution" award. Again the maximum length for the paper was 2 pages and the rules required us to create a video of the work.[3]

The implemented 3DUI (Figure 17a) was a hybrid interface that mixed regular 2D interaction of Blender with spatial 3D interaction developed by the author: the position and orientation of the PlayStation Move controller steered a 3D cursor that could be used to draw metaballs, move and rotate objects or their sub-components, extrude polygons of a mesh, and duplicate objects. Moreover, the PlayStation Move controller could also be employed as an emulated 2D mouse for selection and axis-constrained manipulation, as well as projective texture painting. Thus the 3DUI allowed the use of Blender solely via PlayStation Move and Navigation controllers. Despite having several non-trivial features, the 3DUI was developed in a short time of 3 weeks by the author alone.



(a) (b)

Figure 17. (a) 3DUI for Blender. (b) Professional and novice participants' beliefs regarding future utilization of immersive technology in 3D modelling.

Publication [P5] included a user study with 7 professional 3D artists and 7 novices who had little or no experience with 3D modeling. The task for the participants was to build a 3D model of a castle with the 3DUI, based on a reference image. The study gauged the participants' beliefs on how common the use of 3D input devices, stereo 3D, and head-tracked motion parallax will become among professional 3D modelers in the next 10 years. After completing the task half of the participants believed that in the future 3D displays will dominate over 2D displays in terms of usage, whereas a majority thought that 3D input devices and head tracking will be common, but their use will be secondary to standard interfaces with 2D input and no head tracking (Figure 17b).

[3] 3DUI Contest 2013 Blender entry video: https://youtu.be/mF0ioY7ctkM

The participants also answered open questions about the good and bad sides of the 3DUI: good depth perception, fun, intuitiveness, fatigue (mostly eye related), and poor controller accuracy stood out as frequently mentioned themes. Like in the case of our 3DUI puzzle study from publication [P6], the aforementioned findings suggest that an immersive 3DUI could offer some benefits for 3D modelling as well, but again with possible caveats.

Furthermore, the participants used a Likert scale to rate the 3DUI in terms of fun, intuitiveness, ease of use, and subjective performance. The only statistically significant difference between the two participant groups was that the professionals considered the 3DUI to be more fun than the novices.

It is our opinion that the quality of the 3D models created by the study participants, the short development time of 3 weeks, and the 3DUI's many features demonstrated well how much can be accomplished with affordable, off-the-shelf equipment and open source software. This ties in with the user-led innovation theme that was discussed in Section 2.3, and underlines the potential of easily available technology.

5.4 TurboTuscany VR Experience

We introduced TurboTuscany VR demo in a one-page abstract that was published in IEEE VR 2014 conference's video track [110], along with an associated presentation video.[4] The demo showcased features of RUIS for Unity in action, including the MBC prefab and its hybrid locomotion interface: Kinect full-body tracking was utilized simultaneously with traditional, controller-based 3D character locomotion.

Figure 18. User's view of their virtual avatar (left), which is tracked with Kinect (right).

[4] IEEE VR 2014 video of TurboTuscany: https://youtu.be/wMEaJWsowfQ

Released in August 2013, TurboTuscany was the first publicly available demo to combine immersive 1st person view of Oculus Rift DK1 to a Kinect-tracked virtual body (Figure 18). TurboTuscany also supported Razer Hydra and PlayStation Move controllers for more accurate positional head tracking and 6DOF manipulation of objects. The absolute orientation tracking of those controllers was employed to correct the yaw angle of Oculus Rift DK1 that was slowly drifting over time, which is another useful feature of the MBC prefab, meant for those head-mounted displays that suffer from yaw drift.

No formal user studies were conducted with TurboTuscany, but the feedback from users who downloaded the demo or tried it in our laboratory was enthusiastic:

- "*I really like the virtual body, it reduces nausea a great deal.*"
- "*I was just practicing tai chi with my virtual body on the rooftop listening to the sound of the wind, and taking in the scenery. The Kinect tracking is near perfect and tracks the entire body! So immersive.*"
- "*It is amazing how much immersion is added with positional tracking and by being able to see your arms and feet in VR. You would not believe how long I spent crouching to look at things close up and swaying from side to side to see around barrels. Fantastic work on this!*"

6. Discussion

The central focus of this thesis, research question 1, was confronted in Section 4.2, where we applied statistical analysis on development difficulty statement ratings from a total of 132 VR developers. In this section we discuss the most severe development challenges that were pointed out by our analysis and related thesis publications, and propose ways to alleviate them, thus addressing research question 2 beyond what was presented in Chapter 3. Research question 3 and future directions of our work will also be examined.

The results of our statistical analysis indicated that the most severe difficulties in VR development are testing of the user interface without VR equipment (B6), and the constant cycle of testing and re-implementation of the VR application program (B5). Both of these difficulties are inherently connected to the VR application program's 3DUI and its evaluation.

VR course students in particular suffered from limited access to VR equipment and its impact on testing. Conversely, in separate surveys both Lahtinen et al. [30] and Tan et al. [31] found that access to computers/networks was the least serious practical issue for programming course students. We believe that this issue of limited access to VR equipment will gradually become less pronounced, when consumer VR products become more common. With the exception of very rare or expensive VR systems, this problem should solve itself and does not necessarily require intervention.

The inconvenience caused by the constant cycle of testing and re-implementation in VR development is not easy to tackle. In the future this could be partially addressed with automated 3DUI testing that bears resemblance to automated 2D user interface testing. In such a scheme recorded or simulated user actions would be automatically applied to the application program during testing, sparing the developers from performing the actions themselves.

In our experience the cycle of testing and re-implementation is exacerbated by the following, common 3DUI issues:

I. **Implementation of 3D interaction is required**: the desired interaction behavior does not exist as a reusable component and needs to be implemented from scratch.

II. **No clear mental image of the desired 3D interaction**: this can occur when exploring new or complex interaction styles, or in those cases where the developer lacks 3DUI experience.

III. **Unexpected 3D interaction side-effects**: the interaction works mostly as intended, but introduces unexpected and unwanted side-effects.

Publication [P2] introduced evidence about the prevalence of issue I in 3DUI application program development, particularly the high implementation rate of navigation and object manipulation techniques. Issues II and III reflect our own experiences, what we have witnessed in the students of our VR course, and the challenges described by Wingrave and LaViola [7].

We argue that the above issues and the constant cycle of testing and re-implementation will be alleviated with the emergence of widely-accepted 3D interaction standards and VR toolkits that provide high-level 3DUI building blocks. When VR and AR application programs become more prevalent, 3DUI conventions and standards should surface, as has happened with WIMP and multi-touch interfaces. That should harmonize the 3D interaction expectations of general public and developers to some extent, increase their knowledge about 3DUIs, and consequently mitigate aforementioned issues II and III.

3DUI building blocks address issue I head-on, and issues II and III indirectly, because over time VR developers will learn to understand the interaction capabilities of their toolkit's building blocks and the associated side-effects. Ideally VR toolkits would offer versatile and robust 3DUI building blocks that could be modified and combined in a way that allows a multitude of meaningful 3D interaction behaviors to be created.

The statement regarding difficulties caused by a lack of proper 3DUI building blocks (B3) was rated significantly less severe than statements B5 and B6, and significantly more severe than A5 and A9. There were no statistically significant differences in pairwise comparisons with the other 13 difficulty statements, and the lack of 3DUI building blocks stands somewhere in the mid-range when it comes to development difficulty severity. Nevertheless, we speculate that 3DUI building blocks will be one of the most important means for facilitating VR development.

We assert that skillfully crafted, useful 3DUI building blocks would be adopted by developers for regular use. This would tackle the issue of developers frequently implementing basic 3DUI features rather than inheriting them, saving time and effort, while allowing developers to work on a higher level of abstraction. A VR toolkit with such 3DUI building blocks could potentially hasten the emergence of 3DUI standards, if it was used to create massively popular VR application programs.

The RUIS toolkit and the low barrier of entry requirements for a VR toolkit (R1-R8 in Chapter 3) that it fulfills are our attempt to facilitate VR development and address the research question 2. RUIS for Unity also features our initial effort of providing high-level 3DUI building blocks, which have eased some areas of VR application program development for us and our students.

Currently the most prominent 3DUI building blocks in RUIS are the MBC prefab with its humanoid avatar posing and navigation capabilities, the wand prefabs with their different adjustable selection and manipulation behaviors,

and the selectable object templates. They are part of our preliminary exploration on how 3DUI building blocks can function in a visual environment like Unity Editor. We expect that such 3DUI building blocks will become increasingly useful for implementing typical 3DUI concepts: selection, manipulation, interactive objects, wayfinding, locomotion, gesture controls, and 3D widgets.

Looking at Section 2.4.1 and the sheer number of papers dedicated to different approaches in reusable components for building 3D interaction techniques, it appears that we are not alone in having high hopes for 3DUI building blocks of some kind.

Why then these existing approaches to reusable 3DUI components have not become widely adopted among VR and 3DUI developers? We speculate that this is due to similar reasons as why there is so much fragmentation in VR toolkits; continuous updates are required for the 3DUI building blocks themselves or the underlying hardware abstraction layer to maintain compatibility with the constantly evolving VR hardware and software. Furthermore, the 3DUI developer community has been relatively small, and the effort towards creating reusable 3DUI building blocks has mostly come from researchers, and possibly with other researchers in mind as the target audience. Perhaps this worldwide community has not yet reached a "critical mass" of developers or external attention that would support the continuous development of such high-level tools.

VR industry insiders have estimated that currently Unity 3D is used to create a vast majority of VR application programs [111, 112]. It is likely that some kind of 3DUI building blocks will become widely utilized on popular development suites such as Unity 3D or Unreal Engine, which already have a critical mass of developers. Both development suites have been steadily furnished with VR development features and native support for different HMDs by their respective companies, so it is conceivable that they will one day provide 3DUI building blocks like they are now providing components for creating 2D interfaces.

The company behind Leap Motion finger tracking device has released Unity components and prefabs for implementing gesture-based interaction techniques, 3D widgets, and other 3DUI elements [113, 114]. These are similar to the 3DUI building blocks in RUIS for Unity, and referred to with matching vocabulary; for example, the pinch utility is described as a "*fundamental interactive 'building block.'*" [114]

Third party developers have recently developed VR toolkits with a rich set of 3DUI building blocks for HTC Vive, two of the most notable toolkits being Newton VR [103] and VRTK [104] for Unity 3D. Regardless of who creates reusable 3D interaction technique components, the existing literature on the subject can guide their work.

The 3DUI building blocks in RUIS for Unity have not been formally studied; we merely have presented them in this thesis. VR application programs that utilized our 3DUI building blocks were examined mostly qualitatively in publications [P1], [P3], and [110], which gives some idea of their capabilities.

However, this thesis does also contain statistical evidence on how RUIS can help when compared to other toolkits, as presented in publications [P1] and [P2]; the former demonstrated how our students who had used RUIS for Unity rated the difficulty regarding the lack of 3DUI building blocks to be significantly less severe than a group of developers using other high-level VR toolkits. Publication [P3] showed how we were able to improve RUIS by porting it from Processing to Unity, resulting in significantly reduced development difficulties, more complex VR application programs, and more willingness from students to showcase them.

A student of our 2015 VR course gave the following feedback that aptly describes the features present in our toolkit: "*RUIS was useful, but it abstracted away all the technical details: I can't use Move / Kinect / Oculus Rift without it.*" Even though the abstraction offered by RUIS was in conflict with this particular student's personal learning goals, our VR toolkit performed its role just as intended; it provided useful, high-level means to utilize VR equipment.

Developers using RUIS are expected to have basic skills with Unity development environment, which is relatively easy to learn but still takes some time. If a developer wants to include features beyond what RUIS offers, they need to learn C# programming language. VR development could be made even more accessible by creating a VR toolkit that can be used via a visual programming language, such as the one used in Alice programming environment [115] or the Blueprints Visual Scripting of Unreal Engine. Such attempts have been made by individual 3DUI researchers, as we discussed in Section 2.4.1, but so far these existing 3DUI frameworks have not become popular among developers.

Visual programming or easier scripting languages in software toolkits might help inexperienced 3DUI developers in particular, who according to the results of publication [P2] had significantly more difficulty with the programming interface of their 3DUI toolkit (A8) and programming in general (A9), when compared to experienced developers. Providing developers with good tutorials about the toolkit should also alleviate the former difficulty.

Our analysis of the 132 VR developers in Section 4.2 revealed that from the early obstacles in VR development, the most severe were the high price of the output devices (A2) and the lacking VR toolkit documentation (A6). The former obstacle should diminish over time when the prices of consumer VR products come down. As for the latter obstacle, VR toolkit authors can and should address the lack of documentation and tutorials.

Publications [P4–P6] showcased the flexibility of RUIS as a VR toolkit that can be used to implement different kinds of VR application programs from various application domains and with assorted VR hardware. They also presented user studies that explored the benefits of immersive VR technology for software users, contributing to and strengthening the existing knowledge related to research question 3. Fun and intuitiveness of VR experiences stood out in publications [P5] and [P6], whereas the results of publication [P4] suggested that standard VR depth cues such as stereo 3D and head-tracked motion parallax improve the user experience and work best with non-distracting

lightning. The evidence related to improved user experience is in line with earlier research [116, 117], as is the evidence about fun and intuitiveness [118, 119].

We share the concern of Andujar and Brunet [15] that some of our user study findings – like in many similar studies in the field – might have limited applicability due to the their very specific hardware and task conditions.

From a goal-oriented viewpoint it might have been better to first collect quantitative data about VR development challenges, and then offer solutions to them in the form of a VR toolkit. However, we first formulated the fundamental principles of RUIS toolkit utilizing our own experiences and existing VR literature, and only after development on RUIS had started, we created the 3DUI questionnaire and its development difficulty statements. Further RUIS development and gathering of quantitative data about the difficulties occurred side-by-side over the years. The transition from Processing to Unity was our intervention to address some of the VR development challenges experienced by us and our students when using Processing.

We utilized our difficulty-based benchmark to compare different versions of RUIS and other toolkits. The comparisons performed in publications [P1] and [P2] both involved juxtaposing a relatively homogeneous group of students who utilized RUIS with another, more heterogeneous group utilizing other toolkits. The students of the former group all answered the 3DUI questionnaire as part of our VR course, while the latter group were developers from around the world with different backgrounds. Comparing groups with such varied compositions is not ideal, but our conjecture in both publications was that the more heterogeneous group represented a typical developer using the evaluated toolkits, at least on average.

Conversely, the compared groups of students using different RUIS versions in publication [P3] were very similar: they all attended our VR course, their demographics were closely matched, and all but two students created entertainment VR application programs. This student homogeneity and the possibility for a controlled setting are the reasons why in Section 4.4.2 we suggest that a software project course can act as an environment for studying different factors in software development.

In publication [P2] we compared the different strategies adopted by Microsoft with its Kinect and Sony with its PlayStation Move devices, and suggested how producer companies could encourage user innovation. More general guidelines can be found in the user-led innovation literature: Von Hippel et al. listed specific actions that companies can do to embrace user innovation. For example, user innovators should be credited if a company adopts their designs [34]. Gault and von Hippel presented a survey showing that a significant fraction of user-innovators are freely sharing their innovations in an open source manner, and suggested updates to intellectual property rights policies so that they would better facilitate innovation transfer [120].

6.1 Future Directions

With the recent release of Oculus Rift CV1 and HTC Vive there are a plethora of new VR games released by mostly small independent developers. Within a year or two we should be able to say whether VR games will be the first killer application of VR, or whether they remain as a niche market.

The initial sales of these new consumer HMDs look promising, and there appears to be more VR developers than ever before, encouraged by the growing hype that has surrounded VR ever since Oculus Rift DK1 was released. While this new generation of VR developers will most likely reinvent the wheel many times over with respect to existing VR literature, we remain confident that the increasing pool of aspiring VR developers and hobbyists will result in increased user-led innovation and possibly even major breakthroughs or killer applications. So far the new VR developers have been experimenting copiously with novel forms of locomotion that minimize nausea: e.g. different forms of teleporting, portals, vehicle cockpits as static frame of reference, and limiting field of view.[5] We expect that some of these developers will direct their energy into creating 3DUI building blocks of some form.

Using the lessons learned from our research, we plan to update RUIS for Unity so that some of the identified developer issues will be alleviated. First and foremost, we intend to address the low reuse of 3D interaction techniques by significantly increasing the number and functionality of the 3DUI building blocks in RUIS. The most straightforward way to achieve this is to integrate VRTK [104] into RUIS, which has quickly become popular among developers and entails a permissive software license. The improved 3DUI building blocks should also ease the constant cycle of testing and re-implementation in the development, as discussed in Section 6. Additionally, we plan to extend the documentation of RUIS for Unity and add tutorials, since our research indicated that the lack of toolkit documentation is one of the most severe early obstacles in 3DUI development.

As we have discussed in this thesis, the field of VR has already benefitted from user-led innovation, but VR as a topic has largely remained untouched by the user innovation research community. More research is warranted on user-led innovation in VR, which presents an opportunity to researchers of that field.

The ongoing surge in new VR developers provides us with a good set of circumstances for collecting more data via our 3DUI questionnaire. Based on the thesis author's recent interactions with new VR developers in hackathons and online communities, many of them have some background in game development. The challenges faced by them could differ from those of the VR developers presented in this thesis.

While some in the new generation of developers are eager to consume VR research literature and learn precise terminology, others are not so familiar with academic definitions of 3DUI, AR, and related terms. Thus, the use of technical jargon within the rapidly expanding VR and AR communities could be-

[5] Locomotion in VR: https://youtu.be/p0YxzgQG2-E

come muddled. We are considering updating our 3DUI questionnaire so that it is more approachable for hobbyist VR and AR developers, who might be misled or confused by the terminology. For example, a description such as "*3DUI that utilizes 3D spatial input devices*" is not necessarily understood by VR hobbyists.

For the purposes of our questionnaire the definition of 3DUI is too loose, because it also allows traditional 3D software that is used only via a mouse, a keyboard, and 2D display, as long as the interaction happens in a 3D context. Since we are interested in 3DUI application programs with spatial 3D interaction and immersive displays, we had to make the questionnaire introduction fairly lengthy in order to describe requirements for potential participants.

One limitation of our research is that the 3DUI questionnaire was answered by mostly Western developers, with only a handful of participants being from Asia. This failure to reach Asian developers could possibly mean that some 3DUI development related aspects that are relevant in Asia have been overlooked by our research. In the future this will be addressed by trying to enroll participants more evenly around the world, and possibly by creating translated versions of the 3DUI questionnaire.

Our study of development challenges contains the weakness that our predefined difficulty statements A1-A10 and B1-B8 might not cover all the relevant issues, potentially missing difficulties that we have not thought of. This can be remedied by adding new statements when unaddressed challenges are discovered, either through literature review or analysis of answers to open-ended questions about development challenges. Such analysis was demonstrated in Section 4.2.1, which indicated that our difficulty statements covered extensively the most frequently mentioned issues experienced by the participants of our 3DUI questionnaire. Each statement could also be re-evaluated and modified in cases where the statement wording is suspected to cause avoidable bias in the ratings.

Some participants to our 3DUI questionnaire gave feedback that the questionnaire was rather long, as it easily takes between 20 to 30 minutes of time to fill. The long length can induce survey fatigue and degrade the quality of the answers. This could be alleviated by removing certain parts of the questionnaire based on the study focus; for example, if we only wanted more data on development difficulties regardless of the developer background or details about their application programs, then the other questions could be dropped.

Our approach of using surveys to gather information about developers, VR application programs, and the use of VR toolkits [P2] is time consuming both for the participants and researchers. A more efficient approach could be if the development suite – such as Unity 3D or Unreal Engine – would collect some of this information automatically. For example, data regarding the use of different VR features on the code level (e.g. invocations to methods and variables directly related to HMD or 6DOF controllers) could be collected during compilation time and sent to the analytics team of the development suite. Such data could provide further insight on VR developers and their challenges.

7. Summary

This thesis presented a holistic view on VR software; firstly, we grounded our work on existing literature about VR, software engineering, and user-led innovation. Secondly, we surveyed and analyzed VR developers, their challenges, and development tools extensively. Thirdly, we also assessed VR application programs and their users' experiences in case studies.

The research at hand was largely based on our 3DUI questionnaire that was developed as part of our undertaking to collect data about VR and 3DUI developers. This data was the foundation for our investigation of various aspects in 3DUI development, and we used it to outlined the general demographics of 3DUI developers, the software and hardware that they employed, and the application programs that they created.

As far as we know, our work represents the first attempt to quantitatively analyze difficulties in VR development by utilizing survey methodology. Research question 1 of the thesis asked what the most severe challenges in VR software development are. We addressed this question by investigating difficulty ratings from 132 VR developers, which produced the most important empirical results of the thesis: a ranking of development difficulties based on their severity, accompanied with a list of statistically significant pairwise differences. This ranking has implications on how to facilitate VR development and alleviate its challenges, which could also potentially affect AR and MR development.

We identified the constant cycle of testing and re-implementation in VR as the most serious challenge that could benefit from intervention. Our questionnaire data also revealed other interesting findings, the foremost one being developers' tendency to implement basic 3D interaction techniques instead of inheriting them from VR toolkits. Solutions to these and other development challenges were proposed. We particularly emphasized the possible advantages of high-level 3DUI building blocks, which function as reusable components that can be modified and combined.

The main effort in the work at hand has gone into the technical development of the RUIS toolkit, which among the results of this thesis is the most tangible one. We described how RUIS and the principles behind it are our endeavor to facilitate VR development and make it easier to hobbyists, who we believe to have considerable – yet so far mostly untapped – potential for producing user innovations in the field of VR. Thus, with RUIS toolkit and its principles we have addressed research question 2, which inquired how to facilitate VR soft-

ware development and address its challenges. We also directly confronted research question 2 with our proposals to alleviate the most severe development challenges in Section 6.

In our research we have considered students of VR courses as hobbyists of VR. We presented a detailed account of our VR course and shared the lessons learned during the five years that we organized it, introducing the first longitudinal study where several VR course iterations were extensively analysed.

This thesis introduced new methods for benchmarking VR toolkits, which help to compare toolkits by specific characteristics. The use of these benchmarks was demonstrated via examples; comparisons between our students, who had utilized different RUIS versions in three successive years, contributed knowledge on the different ways in which the chosen VR toolkit can affect the created VR application program and the related development experience.

The thesis also presented several VR application programs of varying nature, which were created by us and our students with RUIS, demonstrating the flexibility of RUIS for VR development. Three of our VR application programs were part of user studies, where we explored lighting-based depth cues and provided evidence that increased fun and intuitiveness can be some of the advantages of 3DUIs, when compared to traditional user interfaces. These findings addressed research question 3, which asked what benefits immersive VR offers in contrast to traditional application programs.

Most of all, we enjoyed the raw creativity in our students' VR application programs and the knowledge that we had been able to provide them with the means to construct such creations.

References

[1] B. Lang, “At the End of 2014, Tech Giants Sony, Facebook, Google, Samsung, Apple, and Microsoft All Have a Hand in VR”, Road to VR, 2014. http://www.roadtovr.com/end-2014-tech-giants-sony-facebook-samsung-google-apple-microsoft-hand-vr/ (accessed February 16, 2015).

[2] M. Rose, “Oculus Rift has sold more than 100,000 dev kits”, Gamasutra, 2014. http://gamasutra.com/view/news/220125/Oculus_Rift_has_sold_more_than_100000_dev_kits.php (accessed December 8, 2014).

[3] “App”, STVR, 2016. http://stv.re/category/app/ (accessed June 12, 2016).

[4] D. Pearson, “VR needs a killer app, not games, to become mainstream”, GamesIndustry.biz, 2014. http://www.gamesindustry.biz/articles/2014-11-12-vr-needs-a-killer-app-not-games-to-become-mainstream (accessed February 18, 2015).

[5] D.A. Bowman, E. Kruijff, J.J. LaViola Jr, I. Poupyrev, “3D user interfaces: theory and practice”, Addison-Wesley, 2004.

[6] M. Green, R. Jacob, “SIGGRAPH ’90 workshop report: Software architectures and metaphors for non-WIMP user interfaces”, Computer Graphics, 25 (3), 1991, pp. 229–235.

[7] C.A. Wingrave, J.J. LaViola Jr, “Reflecting on the design and implementation issues of virtual environments”, Presence: Teleoperators and Virtual Environments, 19 (2), 2010, pp. 179–195.

[8] A. Steed, “Some useful abstractions for re-usable virtual environment platforms”, Software Engineering and Architectures for Realtime Interactive Systems-SEARIS, 2008.

[9] G.D. Crnkovic, “Constructive research and info-computational knowledge generation”, in: Model-Based Reasoning in Science and Technology, Springer, 2010, pp. 359–380.

[10] W.R. Sherman, A.B. Craig, “Understanding virtual reality: Interface, application, and design”, Elsevier, 2002.

[11] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, “Surround-screen projection-based virtual reality: the design and implementation of the CAVE”, in: Proc. Conference on Computer Graphics and Interactive Techniques, 1993, pp. 135–142. http://dl.acm.org/citation.cfm?id=166117.166134 (accessed September 11, 2012).

[12] T. Mazuryk, M. Gervautz, “Virtual reality-history, applications, technology and future”, 1996.

[13] J.J. LaViola Jr, “Bringing VR and spatial 3D interaction to the masses through video games”, Computer Graphics and Applications, IEEE, 28 (5), 2008, pp. 10–15.

[14] I.E. Sutherland, “A head-mounted three dimensional display”, in: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, ACM, 1968, pp. 757–764.

[15] C. Andujar, P. Brunet, “A Critical Analysis of Human-Subject Experiments in Virtual Reality and 3D User Interfaces”, in: Virtual Realities, Springer, 2015, pp. 79–90.

[16] SEARIS, “Software Engineering and Architectures for Realtime Interactive Systems Working Group”, 2016. http://www.searis.net/ (accessed April 6, 2016).

[17] R. Pressman, B.R. Maxim, “Software Engineering: A Practitioner’s Approach”, 8th ed., McGraw-Hill, Inc., New York, NY, USA, 2015.
[18] R.M. Taylor II, T.C. Hudson, A. Seeger, H. Weber, J. Juliano, A.T. Helser, “VRPN: a device-independent, network-transparent VR peripheral system”, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, ACM, 2001, pp. 55–61.
[19] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, C. Cruz-Neira, “VR Juggler: A virtual platform for virtual reality application development”, in: Virtual Reality, 2001. Proceedings. IEEE, IEEE, 2001, pp. 89–96.
[20] WorldViz LLC, “Vizard virtual reality toolkit”, 2016. http://www.worldviz.com/vizard-virtual-reality-software/ (accessed April 5, 2016).
[21] G.J. Kim, K.C. Kang, H. Kim, J. Lee, “Software Engineering of Virtual Worlds”, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, ACM, New York, NY, USA, 1998, pp. 131–138. doi:10.1145/293701.293718.
[22] V. Tanriverdi, R.J. Jacob, “VRID: a design model and methodology for developing virtual reality interfaces”, in: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, ACM, 2001, pp. 175–182.
[23] K. Kaur, “Designing Virtual Environments for Usability”, PhD Thesis, City University, 1998.
[24] C. Fencott, “Towards a design methodology for virtual environments”, in: Workshop on User Centered Design and Implementation of Virtual Environments. University of York, Citeseer, 1999.
[25] F. Mattioli, D. Caetano, A. Cardoso, E. Lamounier, “On the Agile Development of Virtual Reality Systems”, in: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens, 2015, pp. 10–16.
[26] B.A. Myers, “Why are human-computer interfaces difficult to design and implement”, Carnegie Mellon University, Pittsburgh, PA, 1993, Technical Report CMU-CS-93-183.
[27] M. Csisinko, H. Kaufmann, “VITAL - The virtual environment interaction technique abstraction layer”, in: Proceedings of the IEEE Virtual Reality SEARIS Workshop, Waltham, MA, USA, 2010, pp. 77–86.
[28] B. Pellens, F. Kleinermann, O. De Troyer, “An Approach Facilitating 3D/VR System Development Using Behavior Design Patterns”, in: Proceedings of the IEEE Virtual Reality SEARIS Workshop, Waltham, MA, USA, 2010.
[29] T. Duval, A. Blouin, J.-M. Jézéquel, “When Model Driven Engineering meets virtual reality: Feedback from application to the Collaviz framework”, in: Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2014 IEEE 7th Workshop on, IEEE, 2014, pp. 27–34.
[30] E. Lahtinen, K. Ala-Mutka, H.-M. Järvinen, “A Study of the Difficulties of Novice Programmers”, SIGCSE Bull., 37 (3), 2005, pp. 14–18. doi:10.1145/1151954.1067453.
[31] P.-H. Tan, C.-Y. Ting, S.-W. Ling, “Learning Difficulties in Programming Courses: Undergraduates’ Perspective and Perception”, in: International Conference on Computer Technology and Development, IEEE, 2009, pp. 42–46. doi:10.1109/ICCTD.2009.188.
[32] J.C. Lee, “Hacking the Nintendo Wii Remote”, IEEE Pervasive Computing, 7 (3), 2008, pp. 39–45. doi:10.1109/MPRV.2008.53.
[33] E. Von Hippel, “Democratizing innovation”, MIT Press, Cambridge, Mass., 2005.
[34] E. Von Hippel, S. Ogawa, J.P. De Jong, “The age of the consumer-innovator”, MIT Sloan Management Review, 53 (1), 2011, pp. 27–35.

[35] A. Gambardella, C. Raasch, E.A. Von Hippel, “The user innovation paradigm: impacts on markets and welfare”, Social Science Electronic Publishing, Inc, 2015. doi:10.2139/ssrn.2079763.
[36] T. Watts, G. Swann, N.R. Pandit, “Virtual reality and innovation potential”, Business Strategy Review, 9 (3), 1998, pp. 45–54.
[37] J. Whyte, “Innovation and users: virtual reality in the construction sector”, Construction Management and Economics, 21 (6), 2003, pp. 565–572.
[38] Y. Aoyama, H. Izushi, “User-led innovation and the video game industry”, in: Submitted to IRP Conference, London, 2008.
[39] R. Pausch, “Virtual Reality on Five Dollars a Day”, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, New York, NY, USA, 1991, pp. 265–270. doi:10.1145/108844.108913.
[40] A. Robertson, M. Zelenko, “The Rise and Fall and Rise of Virtual Reality”, The Verge, 2014. http://www.theverge.com/a/virtual-reality/oral_history (accessed March 25, 2016).
[41] N. Lavroff, “Virtual Reality Playhouse”, Waite Group Press, 1992.
[42] L. Jacobson, “Garage Virtual Reality”, Sams, 1994.
[43] R. Hollands, “The Virtual Reality Homebrewer’s Handbook”, Wiley, 1996.
[44] K. Pimentel, K. Teixeira, “Virtual Reality: through the new looking glass”, Windcrest Books, 1993.
[45] G.S. Weber, “The New Medium of Expression: Introducing Virtual Reality and Anticipating Copyright Issues”, The John Marshall Journal of Information Technology & Privacy Law, 12 (2), 1993, pp. 175–194.
[46] L. Rebenitsch, “Managing Cybersickness in Virtual Reality”, XRDS, 22 (1), 2015, pp. 46–51. doi:10.1145/2810054.
[47] T. Schou, H.J. Gardner, “A Wii remote, a game engine, five sensor bars and a virtual reality theatre”, in: Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces, ACM, 2007, pp. 231–234.
[48] L. Gallo, G. De Pietro, I. Marra, “3D interaction with volumetric medical data: experiencing the Wiimote”, in: Proceedings of the 1st International Conference on Ambient Media and Systems, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 14.
[49] B. Williamson, C. Wingrave, J.J. LaViola Jr, “RealNav: Exploring natural user interfaces for locomotion in video games”, in: 3D User Interfaces (3DUI), 2010 IEEE Symposium on, IEEE, 2010, pp. 3–10.
[50] J. Giles, “Inside the race to hack the Kinect”, New Scientist, 208 (2789), 2010, pp. 22–23.
[51] D. Mosher, “Six Months Later, Kinect Hacks Flourish”, Wired, 2011. http://www.wired.com/2011/05/johnny-lee-kinect-hacking/ (accessed March 4, 2016).
[52] A. Basu, C. Saupe, E. Refour, A. Raij, K. Johnsen, “Immersive 3DUI on one dollar a day”, in: 2012 IEEE Symposium on 3D User Interfaces (3DUI), 2012, pp. 97–100. doi:10.1109/3DUI.2012.6184191.
[53] P. Wins, A. Basu, K. Johnsen, “Off-the-Shelf Electronics Prototyping for Virtual Reality”, in: IEEE VR Workshop on Off-the-Shelf Virtual Reality, 2012.
[54] P. Wins, A. Basu, K. Johnsen, “Do-It-Yourself Interface Device Prototyping for Virtual Reality”, The International Journal of Virtual Reality, 11 (3), 2012, pp. 43–48.
[55] N. Schneider, “MTBS Forum”, 2007. http://www.mtbs3d.com/phpbb/index.php (accessed February 23, 2016).
[56] D. Woligroski, “3D, Virtual Reality, And Immersive Technology At The U Of OIT”, Tom’s Hardware, 2013.

http://www.tomshardware.com/reviews/augmented-reality-immersive-technology-university-of-ontario,3630.html (accessed February 23, 2016).
[57] L. Palmer, “PR2, a DIY, low cost, high FOV stereoscopic HMD”, MTBS Forum, 2011. http://www.mtbs3d.com/phpBB/viewtopic.php?f=120&t=13745 (accessed March 21, 2016).
[58] D. Nelson, “It’s alive!”, MxR Blog, 2012. http://projects.ict.usc.edu/mxr/blog/it%E2%80%99s-alive/ (accessed March 20, 2016).
[59] M. Bolas, P. Hoberman, T. Phan, P. Luckey, J. Iliff, N. Burba, et al., “Open virtual reality”, in: IEEE Virtual Reality (VR), IEEE, 2013, pp. 183–184.
[60] D.M. Ewalt, “Palmer Luckey: Defying Reality”, Forbes, 2015. http://www.forbes.com/sites/davidewalt/2015/01/05/palmer-luckey-oculus-rift-vr/ (accessed March 20, 2016).
[61] P. Rubin, “The Inside Story of Oculus Rift and How Virtual Reality Became Reality”, Wired, 2014. http://www.wired.com/2014/05/oculus-rift-4/ (accessed March 20, 2016).
[62] E. Carson, “Why virtual reality could finally mend its broken promise”, TechRepublic, 2015. http://www.techrepublic.com/article/why-virtual-reality-could-finally-mend-its-broken-promise/ (accessed March 25, 2016).
[63] A. Wawro, “Oculus VR is funding about two dozen Rift-exclusive games”, Gamasutra, 2015. http://www.gamasutra.com/view/news/247979/Oculus_VR_is_funding_about_two_dozen_Riftexclusive_games (accessed April 3, 2016).
[64] C. Baldwin, E. Von Hippel, “Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation”, Organization Science, 22 (6), 2011, pp. 1399–1417. doi:10.1287/orsc.1100.0618.
[65] E. Von Hippel, R. Katz, “Shifting innovation to users via toolkits”, Management Science, 48 (7), 2002, pp. 821–833.
[66] C. Shaw, M. Green, J. Liang, Y. Sun, “Decoupled simulation in virtual reality with the MR toolkit”, ACM Transactions on Information Systems (TOIS), 11 (3), 1993, pp. 287–317.
[67] C. Carlsson, O. Hagsand, “DIVE A multi-user virtual reality system”, in: Virtual Reality Annual International Symposium, IEEE, 1993, pp. 394–400.
[68] A. Bierbaum, C. Just, “Software tools for virtual reality application development”, in: SIGGRAPH 98 Course 14: Applied Virtual Reality, Orlando, FL, 1998.
[69] R.M. Taylor, J. Jerald, C. VanderKnyff, J. Wendt, D. Borland, D. Marshburn, et al., “Lessons about virtual environment software systems from 20 years of VE building”, Presence: Teleoperators and Virtual Environments, 19 (2), 2010, pp. 162–178.
[70] P.D. Varcholik, J.J. LaViola Jr, C. Hughes, “The Bespoke 3DUI XNA Framework: a low-cost platform for prototyping 3D spatial interfaces in video games”, in: Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, 2009, pp. 55–61.
[71] B. Gilbert, “Valve is solving virtual reality’s input problem”, Engadget, 2015. http://www.engadget.com/2015/03/04/valve-vr-input/ (accessed March 20, 2016).
[72] A. Zeitler, “Survey and Review of Input Libraries, Frameworks, and Toolkits for Interactive Surfaces and Recommendations for the Squidy Interaction Library”, Master’s Thesis, Ludwig Maximilian University of Munich, 2009.
[73] P. Figueroa, S. Gil, R. Oses, J. Toro, C. Rodriguez, C. Benavides, et al., “Visual Programming for Virtual Reality Applications Based on InTml”, SBC, 3 (1), 2012.

[74] P. Figueroa, W.F. Bischof, P. Boulanger, H.J. Hoover, “Efficient comparison of platform alternatives in interactive virtual reality applications”, International Journal of Human-Computer Studies, 62 (1), 2005, pp. 73–103.

[75] K.A. Ritter, C.W. Borst, T.L. Chambers, “Overview and Assessment of Unity Toolkits for Rapid Development of an Educational VR Application”, International Journal for Innovation Education and Research, 3 (7), 2016.

[76] C. Blanchard, S. Burgess, Y. Harvill, J. Lanier, A. Lasko, M. Oberman, et al., “Reality built for two: a virtual reality tool”, ACM SIGGRAPH Computer Graphics, 24 (2), 1990, pp. 35–36.

[77] R.J. Jacob, L. Deligiannidis, S. Morrison, “A software model and specification language for non-WIMP user interfaces”, ACM Transactions on Computer-Human Interaction (TOCHI), 6 (1), 1999, pp. 1–46.

[78] G. de Haan, F.H. Post, “StateStream: a developer-centric approach towards unifying interaction models and architecture”, in: Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems, ACM, 2009, pp. 13–22.

[79] P. Figueroa, M. Green, H.J. Hoover, “InTml: a description language for VR applications”, in: Proceedings of the Seventh International Conference on 3D Web Technology, ACM, 2002, pp. 53–58.

[80] P. Figueroa, W.F. Bischof, P. Boulanger, H.J. Hoover, R. Taylor, “Intml: A data-flow oriented development system for virtual reality applications”, Presence: Teleoperators and Virtual Environments, 17 (5), 2008, pp. 492–511.

[81] A. Ray, D.A. Bowman, “Towards a system for reusable 3D interaction techniques”, in: Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology, ACM, 2007, pp. 187–190.

[82] C. Wingrave, D. Bowman, “Chasm: Bridging description and implementation of 3d interfaces”, in: Proc. of IEEE VR Workshop on New Directions in 3DUIs, Citeseer, 2005, pp. 85–88.

[83] D. Vanacken, J. De Boeck, C. Raymaekers, K. Coninx, “NiMMiT: A notation for modeling multimodal interaction techniques”, in: Proceedings of the First International Conference on Computer Graphics Theory and Applications (GRAPP 2006), INSTICC, 2006, pp. 224–231.

[84] D. Valkov, B. Bolte, G. Bruder, F. Steinicke, “Viargo-a generic virtual reality interaction library”, in: Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012 5th Workshop on, IEEE, 2012, pp. 23–28.

[85] D. Martínez, J.L. Lawson, J.P. Molina, A.S. García, P. González, J. Vanderdonckt, et al., “A framework to develop VR interaction techniques based on openinterface and AFreeCA”, in: Human-Computer Interaction–INTERACT 2011, Springer, 2011, pp. 1–18.

[86] G.D. Kessler, “A framework for interactors in immersive virtual environments”, in: Virtual Reality, 1999. Proceedings., IEEE, IEEE, 1999, pp. 190–197.

[87] C.A. Wingrave, “Concept-Oriented design in Chasm: Conversational domain language inspired 3D user interface design and development”, PhD Thesis, Virginia Polytechnic Institute and State University, 2008.

[88] D.H. Bell, “Teaching virtual reality”, ACM SIGCSE Bulletin, 28 (2), 1996, pp. 56–61. doi:10.1145/228296.228306.

[89] G.W. Zimmerman, D.E. Eber, “When worlds collide!: an interdisciplinary course in virtual-reality art”, in: ACM SIGCSE Bulletin, ACM, 2001, pp. 75–79.

[90] S. Stansfield, “An introductory VR course for undergraduates incorporating foundation, experience and capstone”, in: ACM SIGCSE Bulletin, ACM, 2005, pp. 197–200.

[91] D. Cliburn, “Incorporating Virtual Reality Concepts into the Introductory Computer Graphics Course”, in: Proceedings of the SIGCSE, ACM, Houston, Texas, USA, 2006, pp. 77–81.
[92] B. Herbelin, J. Ciger, “Teaching and learning immersion and presence”, in: 11th Annual International Workshop on Presence, Padova, Italy, 2008, pp. 305–313.
[93] J. Zara, “Virtual Reality course — A natural enrichment of Computer Graphics classes”, in: Computer Graphics Forum, Wiley Online Library, 2006, pp. 105–112.
[94] K. Miyata, K. Umemoto, T. Higuchi, “An educational framework for creating VR application through groupwork”, Computers & Graphics, 34 (6), 2010, pp. 811–819.
[95] P. Häfner, V. Häfner, J. Ovtcharova, “Teaching Methodology for Virtual Reality Practical Course in Engineering Education”, Procedia Computer Science, 25, 2013, pp. 251–260.
[96] G.C. Burdea, “Teaching Virtual Reality: Why and How?”, Presence: Teleoperators and Virtual Environments, 13 (4), 2004, pp. 463–483. doi:10.1162/1054746041944812.
[97] G.C. Burdea, “World-wide Survey of Universities Teaching Virtual Reality”, 2008. http://vrtechnology.org/resources/public/survey.html (accessed April 29, 2016).
[98] R.J. Jacob, A. Girouard, L.M. Hirshfield, M.S. Horn, O. Shaer, E.T. Solovey, et al., “Reality-based interaction: a framework for post-WIMP interfaces”, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2008, pp. 201–210.
[99] C. Reas, B. Fry, “Processing: a programming handbook for visual designers and artists”, Mit Press, 2007, 6812.
[100] T.M. Takala, “Download”, Reality-Based User Interface System, 2016. http://blog.ruisystem.net/download/ (accessed December 5, 2016).
[101] T. Lokki, T. Ilmonen, W. Makela, T. Takala, “Upponurkka: An Inexpensive Immersive Display for Public VR Installations”, in: Virtual Reality Conference, 2006, 2006, pp. 315–315.
[102] M. Matveinen, “The Design and Implementation of a Virtual Reality Toolkit”, Master’s Thesis, Aalto University, 2015. https://aaltodoc.aalto.fi/handle/123456789/16649 (accessed March 20, 2016).
[103] K. Bradner, N. Abel, A. Hunter, “Newton VR”, 2016. https://github.com/TomorrowTodayLabs/NewtonVR (accessed September 10, 2016).
[104] H. Ball, “VRTK”, 2016. http://vrtk.io (accessed September 10, 2016).
[105] T.M. Takala, “Survey: Challenges in VR / 3DUI Development”, 2011. https://docs.google.com/forms/d/1zeelAh4sHY56G1et7pE_g9s0gV7deT47c32X8MW_o9o (accessed December 5, 2016).
[106] A. Field, “Discovering Statistics using IBM SPSS Statistics”, 4th ed., SAGE Publications, 2013.
[107] R. Khlebnikov, B. Kainz, J. Muehl, D. Schmalstieg, “Crepuscular rays for tumor accessibility planning”, Visualization and Computer Graphics, IEEE Transactions on, 17 (12), 2011, pp. 2163–2172.
[108] M.M. Bradley, P.J. Lang, “Measuring emotion: the self-assessment manikin and the semantic differential”, Journal of Behavior Therapy and Experimental Psychiatry, 25 (1), 1994, pp. 49–59.
[109] Blender Foundation, “Home of the Blender project”, 2016. https://www.blender.org/ (accessed May 9, 2016).

[110] T.M. Takala, M. Matveinen, “Full body interaction in virtual reality with affordable hardware”, in: Virtual Reality (VR), 2014 iEEE, IEEE, 2014, pp. 157–157.
[111] J. Gaudiosi, “This company dominates the virtual reality business, and it’s not named Oculus”, Fortune, 2015. http://fortune.com/2015/03/19/unity-virtual-reality/ (accessed June 12, 2016).
[112] P. Graham, “Palmer Luckey: ‘Something like 90% of projects on Gear VR are made using Unity’”, VR Focus, 2016. http://www.vrfocus.com/2016/02/palmer-luckey-something-like-90-of-projects-on-gear-vr-are-made-using-unity/ (accessed June 12, 2016).
[113] A. Colgan, “New Unity Module for User Interface Input”, Leap Motion Blog, 2016. http://blog.leapmotion.com/ui-input-module/ (accessed June 12, 2016).
[114] A. Colgan, “Power at Your Fingertips: Pinch Utilities for Orion”, Leap Motion Blog, 2016. http://blog.leapmotion.com/power-fingertips-pinch-utilities-orion/ (accessed June 12, 2016).
[115] C. Kelleher, D. Cosgrove, D. Culyba, C. Forlines, J. Pratt, R. Pausch, “Alice2: programming without syntax errors”, in: Proceedings of the 15th Annual Symposium on the User Interface Software and Technology, Paris, France, 2002.
[116] J. Takatalo, T. Kawai, J. Kaistinen, G. Nyman, J. Häkkinen, “User experience in 3D stereoscopic games”, Media Psychology, 14 (4), 2011, pp. 387–414.
[117] M.P. Snow, R.C. Williges, “Empirical models based on free-modulus magnitude estimation of perceived presence in virtual environments”, Human Factors: The Journal of the Human Factors and Ergonomics Society, 40 (3), 1998, pp. 386–402.
[118] G. Hackenberg, R. McCall, W. Broll, “Lightweight palm and finger tracking for real-time 3D gesture control”, in: Virtual Reality Conference (VR), 2011 IEEE, IEEE, 2011, pp. 19–26.
[119] W. Hürst, M. Helder, “Mobile 3D graphics and virtual reality interaction”, in: Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, ACM, 2011, p. 28.
[120] F. Gault, E. Von Hippel, “The prevalence of user innovation and free innovation transfers: Implications for statistical indicators and innovation policy”, MIT Sloan Research Paper No. 4722-09, 2009.

This thesis investigates challenges specific to VR software development, and explores methodology for such research. The thesis includes some of the earliest quantitative analysis on VR software development challenges, identifies the most severe development issues, and proposes solutions to them. This has implications on how VR software development could be eased. The analysis is based on data collected from 132 developers of VR application programs, which forms the backbone of the research. The thesis introduces RUIS, a software toolkit for facilitating hobbyist innovation by simplifying the development of VR application programs that rely on immersive displays and spatial interaction devices. Case studies employing VR application programs created with RUIS are included, describing different ways how 3D user interfaces can affect the experience and performance of VR software users.

ISBN 978-952-60-7245-6 (printed)
ISBN 978-952-60-7244-9 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS

Aalto-DD 6/2017