AN OBJECT-ORIENTED SOFTWARE FRAMEWORK
FOR IMMERSIVE VIRTUAL REALITY EXERGAMES

By

Yifan Wang

M.Eng.

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Electrical and Information Engineering

The University of Sydney

March 2020

Faculty of Engineering

The University of Sydney, Australia
Authorship Attribution Statement

Chapter 3, 4 and 5 of this thesis are published as


I designed and developed this framework, extracted and analysed the data and wrote the drafts of the MS.

Chapter 6 of this thesis contains part of the data that published as


I guided the students to develop applications with framework for user studies, co-designed and conducted all the user studies with the corresponding author, wrote part of the drafts of the MS.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Yifan Wang

Signature:

Date: 01/08/2019
Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work.

This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Yifan Wang

Signature:

Date: 01/08/2019
Dedication

To my family and beloved.
Contents

Authorship Attribution Statement ................................................................. ii
Statement of Originality ................................................................................... iv
Dedication .......................................................................................................... v
Contents ........................................................................................................... vi
List of Figures .................................................................................................... x
List of Tables ....................................................................................................... xiv
Acknowledgements .......................................................................................... xv
Preface ................................................................................................................ xvi
Abstract ........................................................................................................... xvii

Chapter 1. Introduction ................................................................................... 1
  1.1 Motivation ................................................................................................. 2
  1.2 Research question ................................................................................... 6
  1.3 Challenges ............................................................................................... 7
  1.4 Contributions .......................................................................................... 8
  1.5 Structure of the thesis ........................................................................... 9

Chapter 2. Background .................................................................................. 11
  2.1 Virtual reality ........................................................................................ 12
    2.1.1 Immersive VR ................................................................................ 14
    2.1.2 VR exergame ................................................................................ 19
    2.1.3 Generalizations of VR exergame design ...................................... 22
    2.1.4 Game engine for VR development ............................................... 24
  2.2 Software framework ............................................................................... 29
    2.2.1 Component-based software engineering ..................................... 31
    2.2.2 Design patterns ............................................................................ 33
## 2.2.3 Object-oriented application framework .................................................. 34

## 2.3 Frameworks for VR exergames ................................................................. 36

2.3.1 Partially immersive frameworks ................................................................. 38

2.3.2 Fully immersive frameworks ................................................................... 47

### Chapter 3.  **Framework Infrastructure** ..................................................... 60

3.1 Introduction ........................................................................................................... 61

3.2 Domain analysis ................................................................................................. 64

3.3 Framework architecture design ........................................................................ 69

3.3.1 Unity game engine ......................................................................................... 70

3.3.2 Proposed architecture ................................................................................. 71

3.4 Framework module design ................................................................................ 73

3.4.1 VR device support module .......................................................................... 74

3.4.2 VR scene module ......................................................................................... 77

3.4.3 Physiological & pedalling device support module ....................................... 79

3.4.4 Middleware module ..................................................................................... 82

3.4.5 User study module ...................................................................................... 83

3.4.6 Database module ......................................................................................... 85

3.5 Framework data flow ....................................................................................... 87

### Chapter 4.  **Framework Implementation** .................................................. 90

4.1 Introduction ........................................................................................................ 91

4.2 Basic class implementation .............................................................................. 93

4.3 Module implementation .................................................................................... 95

4.3.1 VR device support module .......................................................................... 96

4.3.2 VR scene module ......................................................................................... 99

4.3.3 Physiological & pedalling device support module ....................................... 102

4.3.4 Middleware module ..................................................................................... 104

4.3.5 User study module ...................................................................................... 106

vii
Chapter 5. Framework Code Evaluation .......................................................... 112

5.1 Introduction .............................................................................................. 113
5.2 Applications for evaluation ...................................................................... 114
5.3 Framework code efficiency study .............................................................. 118
  5.3.1 Code study 1 ....................................................................................... 118
  5.3.2 Code study 2 ....................................................................................... 120
  5.3.3 Code study 3 ....................................................................................... 122
5.4 Framework code quality analysis ............................................................. 124
  5.4.1 Maintainability index ......................................................................... 124
  5.4.2 Cyclomatic complexity ..................................................................... 126
  5.4.3 Depth of inheritance ......................................................................... 128
  5.4.4 Class coupling ................................................................................. 129
5.5 Discussion ................................................................................................. 130

Chapter 6. Framework Feasibility and Usability Evaluation ............................ 134

6.1 Introduction .............................................................................................. 135
6.2 User studies ............................................................................................. 136
  6.2.1 User study 1 ....................................................................................... 136
  6.2.2 User study 2 ....................................................................................... 138
  6.2.3 User study 3 ....................................................................................... 139
6.3 Framework feasibility evaluation .............................................................. 140
  6.3.1 Success rate ....................................................................................... 141
  6.3.2 Exercise performance ....................................................................... 143
  6.3.3 Motion sickness ............................................................................... 144
  6.3.4 Perceived competence .................................................................... 146
6.4 Framework usability evaluation ............................................................... 148
  6.4.1 Enjoyment ......................................................................................... 149
6.4.2 Immersion .................................................................................................................. 151
6.4.3 Intuitive controls ....................................................................................................... 153
6.4.4 Preference for future play .......................................................................................... 154
6.4.5 Observation ................................................................................................................ 155
6.5 Discussion ..................................................................................................................... 156

Chapter 7. Conclusion ........................................................................................................ 160
7.1 Outcomes ....................................................................................................................... 161
7.2 Future work ...................................................................................................................... 164
7.3 Final conclusion .............................................................................................................. 167

Glossary ............................................................................................................................. 168

Bibliography ....................................................................................................................... 170

Appendices ......................................................................................................................... 187
List of Figures

Figure 2-1. The five classic components of a VR system (reproduced from Burdea & Coiffet, 2003, pp. 13) ................................................................. 12

Figure 2-2. Left – A docking system with haptic device in the work of Hou and Sourina (2011), Right – A user interacting with a model of Earth through KeckCAVES visualization environment (Keck, 2012) ........................................................................ 14

Figure 2-3. The virtual interface environment workstation by NASA (1990) ....................... 15

Figure 2-4. Left - HTC Vive (HTC, 2016), Right - Oculus Rift (Oculus, 2016) ................. 16

Figure 2-5. A language-learning game integrated with Oculus Rift (Cheng et al., 2017) ...... 17

Figure 2-6. A blind user using the simulation system for navigation training in the study of Zhao et al. (2018) .......................................................................................................................... 18

Figure 2-7. Left - A VR cycling exergame from Bolton et al. (2014), Right –Mild cognitive impairment playing the VR rowing exergame reproduced from Eisapour et al., (2018). 21

Figure 2-8. A screenshot of Unity (version 5.6.0f3) ............................................................ 25

Figure 2-9. Scene Window (left) and Game Window (right) in Unity interface, sourced from https://docs.unity3d.com ........................................................................................................ 26

Figure 2-10. Hierarchy Window (left) and Project Window (right) in Unity interface, sourced from https://docs.unity3d.com ........................................................................................................ 27

Figure 2-11. Inspector Window in Unity interface, sourced from https://docs.unity3d.com .. 28

Figure 2-12. Component-based software development, reproduced from Cai et al. (2000) ... 32

Figure 2-13. System architecture of component-based software systems by Cai et al. (2000)33

Figure 2-14. The integrated development environment XVR Studio (Carrozzino et al., 2005) .......................................................................................................................... 39

Figure 2-15. SPRINT system with virtual environment (Ruffaldi et al., 2011) ................. 40

Figure 2-16. Overview of D’Andrea et al. (2013)’s framework ......................................... 41

Figure 2-17. Framework architecture by Avola et al. (2013) ............................................ 42

Figure 2-18. Overall system architecture (Camporesi & Kallmann, 2013) ....................... 44
Figure 2-19. Left - Data flow of framework, Right - Virtual Cycling System built with framework (Kikuchi, 2014) .................................................................46
Figure 2-20. Overview of RUIS architecture (Takala, 2014) ..................................48
Figure 2-21. Immersive VR Exergames developed with RUIS framework (Takala, 2014) ....49
Figure 2-22. Schematic diagram of the proposed framework, reproduced from Paraskevopoulos et al. (2016) ........................................................................................................51
Figure 2-23. Handball goalkeeper using the system to improve performance (Bideau et al., 2010) .....................................................................................................................53
Figure 2-24. Left - Implementation guideline, Right - detailed implementation; reproduced from Shepherd et al. (2018) ..............................................................................54
Figure 2-25. Left - Data capture devices, Right - VR simulator implementation, in Shepherd et al. (2018) ........................................................................................................55
Figure 2-26. Left - Structure of the stroke rehabilitation system, Right - framework design, reproduced from Semblantes et al. (2018) .................................................................56
Figure 2-27. Framework overview (Semblantes et al., 2018).................................57
Figure 3-1. System architecture of our previous VR exergames (Ijaz et al., 2016a, pp. 142).62
Figure 3-2. System setup of our first VR exergame (Ijaz et al., 2016b) ......................63
Figure 3-3. The common architecture of VR exergaming systems which the framework should generate ........................................................................................................68
Figure 3-4. Framework input components ................................................................69
Figure 3-5. Common structure of Unity project .........................................................71
Figure 3-6. Two layered framework architecture ......................................................72
Figure 3-7. VR Device Support Module’s GameObject hierarchy in Unity ...............75
Figure 3-8. Virtual scene provided by this module (Wang et al., 2017) .................77
Figure 3-9. VR Scene Module’s GameObject hierarchy in Unity .............................78
Figure 3-10. Stationary trike (Left) and DeskCycle (Right) ....................................80
Figure 3-11. Microsoft Band 2 (Left) and Fitbit Charge 2 (Right) ............................80
Figure 3-12. Physiological & Pedalling Device Support Module’s GameObject hierarchy in
Figure 3-13. Arduino Uno microcontroller board (Left) and Android phone (Right) .......... 82
Figure 3-14. Middleware Module hierarchy in Unity .................................................. 83
Figure 3-15. User Study Module hierarchy in Unity .................................................... 84
Figure 3-16. Inspector Window of three scripts’ design – PlayerInfo, GameplayParameter and UserStudyManager ................................................................. 84
Figure 3-17. Database Module Design in Unity .............................................................. 85
Figure 3-18. Data flow of the framework in terms of modules ....................................... 88
Figure 4-1. A sample class that is created in Unity Editor and derived from MonoBehaviour ........................................................................................................ 92
Figure 4-2. Simplified order of execution for event functions in Unity, reproduced from (Unity Technologies, 2017) ........................................................................ 93
Figure 4-3. UML class diagram on the relationship among Unity’s base classes (MonoBehaviour and GameObject) and our framework module classes (module behaviour and manager) .................................................................................. 94
Figure 4-4. Main classes in VR Device Support Module ............................................... 97
Figure 4-5. Main classes in VR Scene Module ................................................................. 100
Figure 4-6. Main classes in Physiological & Pedalling Device Support Module ............. 103
Figure 4-7. Main classes in Middleware Module ............................................................. 105
Figure 4-8. Main classes in User Study Module .............................................................. 107
Figure 4-9. Implementation of Database Module ............................................................ 109
Figure 4-10. The implementation of two classes on asset loading ................................. 110
Figure 5-1. Total SLOC provided by developer and framework in each game ............... 119
Figure 5-2. Lines of IL code provided by developer and framework in each game .......... 121
Figure 5-3. Cost comparison of different games ............................................................ 123
Figure 5-4. Maintainability Index analysis result of framework code ............................. 125
Figure 5-5. Cyclomatic Complexity analysis results of framework code ...................... 127
Figure 5-6. Depth of Inheritance analysis results of framework code ........................................ 128
Figure 5-7. Class Coupling analysis results of framework code .............................................. 130
List of Tables

Table 5-1. Detailed investment of each student in project development ........................................... 117
Table 5-2. Total SLOC provided by developer and framework in each game................................. 119
Table 5-3. Lines of IL code provided by developer and framework in each game.......................... 121
Table 6-1. Success rate of participants in user studies................................................................. 141
Table 6-2. Activity time in each study (min:sec)............................................................................. 143
Table 6-3. Number of participants that were likely to have motion sickness and those experienced motion sickness in user studies.......................................................... 145
Table 6-4. Frequency of participants’ responses on the agreement on perceived competence................................................................................................................................. 147
Table 6-5. Frequency of participants’ responses on the agreement on enjoyment ...................... 150
Table 6-6. Frequency of participants’ responses on the agreement on immersion ...................... 151
Table 6-7. Frequency of participants’ responses on the agreement on intuitive controls ......... 153
Table 6-8. Preference for future play in user studies..................................................................... 154
Acknowledgements

I would like to express my sincere appreciation and thanks to my supervisors Prof. Rafael Calvo and Dr. Dong Yuan for guiding me during my PhD and supporting me in all my work.

My special appreciation also goes to Dr. Chunfeng Liu who has greatly helped me on PhD study and my life in Sydney.

I would like to sincerely acknowledge Kiran Ijaz and Johan Alibasa for working with me in projects and helping me in my studies.

My sincere thank also goes to Prof. Richard Ryan, Dr. Naseem Ahmadpour and Dr. David Milne for providing helpful advices to my research and critical evaluations to my papers.

I would particularly like to acknowledge the former and current members of our research group Wellbeing Technology Lab (previously Positive Computing Lab), especially Dorian Peters, A/Prof. Abelardo Pardo, Kathryn McCabe, Muhammad Rabiul Hasan, Simon Hoermann, Yu Zhao, Mingxuan Li, Yunyao Yao, Kaihang Wu, Marlon Santos, Khushnood Naqshbandi, Joshua Yee Haun Kim, Weiyu Ju, Xiao Han and Xiuwen Gong for their kindly encouragement, support and cooperation.

I would also like to acknowledge the financial support of my projects from Charles Perkins Centre, The University of Sydney.

At last, I would specially thank my father, my mother and my beloved for all their supports and patience.
Preface

This thesis is based on the following published peer-reviewed papers:


Abstract

Immersive Virtual Reality (VR) exergames (physical activity through active video games) have significant impacts on research outcomes of human factors. On one hand, VR has positive psychological effects on user experience such as enjoyment and engagement. On the other hand, exergames with VR increase users’ physical activity levels and energy consumption, showing a great potential in improving health. However, the main challenges in employing VR with exergames are the design and development for customised applications targeting different research purposes, which are time consuming and expensive.

In recent years, we have seen a growing number of Human-Computer Interaction (HCI) research projects using frameworks to ease the development and customisation of VR exergames. Nevertheless, these frameworks still face a series of challenges such as how to overcome different design limitations with VR, how to provide a proper guidance of building such frameworks, and how to evaluate the frameworks from different perspectives. Besides, only few projects target on providing an object-oriented framework for VR exergame development.

To overcome the above challenges and to provide a better framework for HCI community, this thesis presents an object-oriented software framework to reduce the effort in building immersive VR exergames. VR design aspects including safety, motion sickness and feedback of activity, are involved in this framework, meanwhile design patterns, modular architecture
and object-oriented programming are used to enable the developer to easily prototype an experiment-oriented VR exergame.

This framework has been used in our research group to develop several VR exergames, where we evaluate how the framework simplifies the development process. Evaluation results show that the development effort and time spent in programming are significantly reduced by using the framework. In addition, the code metrics results of the framework source code are analysed to reveal its good quality in terms of maintainability, reliability and reusability. Further, the framework’s feasibility and usability are also evaluated through several user studies. The user feedback and user performance demonstrate that the VR exergames built with this framework can engage players to do moderate exercise, which shows the practicality of using the framework in experiment-oriented user studies.

The development and customisation of VR applications are important in many domains, not just exergame. The potential impacts of this framework and the discoveries made in this thesis are not just limited to HCI community. The utilization of this framework could also benefit research groups in education, psychology and medicine, who need VR as a tool for research.
Chapter 1. Introduction

Summary

This chapter presents the motivation of this thesis through introducing the VR technology used in HCI research and the existing frameworks used for VR application development in this community. Then the aim of the thesis is described, which is to build an object-oriented software framework for the development of experiment-oriented VR exergames. In addition, the challenges in achieving the aim are also presented together with the statement of contributions of this thesis. The last section overviews the structure of the following chapters.
1.1 Motivation

Virtual Reality (VR) is widely used in different research domains of Human-Computer Interaction (HCI). A variety of studies have suggested that VR can increase the positive psychological effects in exercise (Plante et al., 2003), provide a positive internal learning experience in terms of motivation, enjoyment and engagement in education (Kavanagh et al., 2017), benefit body function, activity and participation outcomes in rehabilitation and therapy (Lohse et al., 2014). In addition, VR is also broadly used in training (Adamovich et al., 2010), simulation and medical applications (Moline, 1997) to improve the quality of health care.

Games are also playing a significant role in HCI’s active areas through research with gamified elements and designs. According to Connolly et al. (2012, pp. 661), games for serious purpose have potential positive impacts on studies such as perception, cognition and motivation. Exercise games, also called exergames for short, are a kind of video games with exercise activity, which not only provide positive psychological outcomes but also promote physical activity for health (Matallaoui et al., 2017, pp. 3317). HCI community has seen a rising number of studies employing VR exergames given that VR improves simulation outcomes (Menin et al., 2018, pp. 57). Example of these scenarios are using VR exergames to enhance older adults’ physical functioning (Molina et al., 2014), applying VR exergames as a practical tool to stimulate rehabilitation activities (Trombetta et al., 2017) and exploiting VR exergames to motivate both children and adults to engage in exercise (Finkelstein & Suma, 2011).

However, most of the popular VR applications are designed for commercial purpose which
targets end-users other than researchers. There is a high demand of VR software applications that are developed for internal research usage. Since research systems usually require a high degree of customisation, e.g. sensor fusion in VR exergame, compared to those provided by commercial systems (Reitmayr & Schmalstieg, 2001, pp. 47), researchers tend to build their own VR exergames.

Nevertheless, developing VR exergames is not an easy job. On one hand, Shaw et al. (2015) argue that there are few major design challenges in VR exergames especially 1) how to solve motion sickness in Virtual Environments (VE), 2) how to provide reliable bodily motion tracking controls in VR and 3) exercise health and safety consideration in VR. On the other hand, VR exergames applied to different areas commonly integrate with various third party hardware devices and sensors to improve user’s experience of game play (Carvalho et al., 2016), which brings complexity in system implementation. Yet customised VR exergames only present partial examples of HCI research, most studies still rely on existing commercial systems and well-controlled studies are still rare (Diest et al., 2013).

Although customised VR software applications are valuable tools for research groups in HCI community, developing software applications is not normally the principal focus of an HCI research group, where the production capacity and code writing skills are usually not equivalent to professional software engineers. However, research facilitators need to afford an appropriate amount of time to ensure the translation of knowledge from other disciplines, e.g. computer science, to HCI community (Olson & Kellogg, 2014) as it brings further advantages.
For instance, though the development of software application requires much effort and time investment, it could lead to a freely available application as a helpful tool used by other researchers, which may also increase the group’s notability (Fdez-Riverola et al., 2012). Furthermore, the real benefits lie in the capability of code reuse that saves resources and efforts for development team.

Not only in VR exergames but also in many software applications, there are repetitive code or shared code that handle basic features such as Graphical User Interface (GUI), user interaction and visualization of results. These are not problem-specific functionalities, but they still require a significant amount of time invested in development even with the core area-related functionalities already implemented. Therefore, in VR exergame development, it is essential to reuse the code that is repetitively shared in the necessary elements (Bouville et al., 2015). Additionally, other problem-specific code including data structure and customised components should also be easily reusable when necessary to improve the productivity of producing applications.

Consequently, in order to enhance the efficiency in building VR exergames, the ability in code reuse is the primary factor that should be taken into consideration. In software engineering, a typical solution of code reuse is to make use of application framework, which is a reusable architecture design in structure that acts as a skeleton of an application in purpose (Johnson, 1997). Bringing together domain-specific design patterns, which is a common vocabulary and best practice for generally occurring problem (Gamma et al., 1994), and software components,
which are units of composition that encapsulate predefined services (Pomberger et al., 1998), Object-Oriented Application Framework (OOAF) is a software technology that offers abstract object-oriented class libraries to simplify the development of software (Fayad & Schmidt, 1997).

With the use of OOAF, the common functionality between VR exergames is provided by the framework, which allows the developer to save time in low level details. Subsequently, they can focus on the development of application-specific features, the higher level of elements that are independent from shared functionalities. This type of code separation improves code reuse and maintenance in both framework usage and problem-specific design (Mitchell, 1990). Essentially according to Fayad & Schmidt (1997), through the reuse of code, design patterns and domain-specific knowledge, OOAF benefits developers with its modularity, reusability extensibility and inversion of control.

Recent years has witnessed the framework getting popular in the development of VR software. Researchers have implemented different domain-specific VR frameworks to produce diverse applications for various HCI studies. Examples are extendable open source framework, AVANGO (Kuck et al., 2008), which uses a component-based design and different patterns to support the development of distributed, interactive virtual reality and augmented reality; PhysioVR (Muñoz et al., 2016), an open source mobile VR framework to ease the integration of wearable devices with physiological sensors in portable virtual reality, targeting mobile affective computing systems and user studies; CalVR (Schulze et al., 2013), an advanced open
source software framework that combines elements from various existing VR middleware to facilitate 3D displays and 3D input devices through its object-oriented class hierarchy; CyberMed (Machado et al., 2009), a free and extendable VR framework to improve the development of medical simulation applications for both new and expertise programmers, by saving the time via its complete synchronization among different functionalities.

However, a thorough search of the relevant literature yielded only few related work on object-oriented framework that included general application framework characteristics meanwhile focused on providing exercise related architecture and experiment-oriented features for HCI studies supporting virtual reality.

1.2 Research question

Given the challenges of designing and developing customised VR exergames, code reuse in programming and the lack of available VR exergaming framework, this thesis aims to build an open source, object-oriented VR framework as a tool that helps research groups to easily prototype VR exergames for studies. To solve the major challenges in VR exergame design referred by Shaw et al. (2015), the framework is able to use optimal design with affordable devices. To enable researchers to customise different exergames, the framework is capable of supporting different devices and hardware. Following are the major research questions addressed in this thesis:

RQ1: What are the design considerations of such an object-oriented application framework for VR exergame?
RQ2: How does the framework benefit researchers in VR exergame development?

RQ3: How to evaluate the quality, feasibility and usability of the framework?

1.3 Challenges

Previous work in the research literature show advances in framework and framework employed in VR, however several challenges must be overcome in order to achieve our aims. Firstly, according to Shaw et al. (2015) and Menin et al. (2018), though virtual reality technology has been broadly used in HCI studies with exergame design, there are still design limitations in VR with Head-Mounted Display (HMD). Secondly, upon the experienced and identified questions during development, there are general activities generated to aid the development of a simple framework (Fayad et al., 1999). However, there is barely specific guidance on how to design and implement such an object-oriented framework in VR exergame domain. Thirdly, literature has revealed the significance of employing framework in software development (Bäumer et al., 1997) and the advantages of using gamification design frameworks in HCI (Mora et al., 2015), there is no consensus on how significant the frameworks are in easing the application development, as well as the degree of effort saving through framework. In other words, there is no standard performance evaluation of such a framework. Finally, if the framework is to be used by other researchers, the framework quality should be feasible and meet the basic requirements known for other real-world code metrics apart from the framework performance.
1.4 Contributions

The contributions of this thesis are interdisciplinary, covering the fields of software engineering and human-computer interaction. The main contributions are summarized as following:

First, the most important contribution is the design and implementation of a useful and open source object-oriented virtual reality application framework, which enables research groups to easily prototype VR exergames without spending too much effort in basic functionalities. The framework modules with its specification are described in the thesis (including the framework architecture, design patterns in framework, module functionality, framework data flow and detail implementation in modules), and they have been successfully embedded into framework design and implementation. This thesis contributes not only an exergaming related design that provides proper exercise feedback in a safe set up without causing motion sickness, but also an architecture embedded with design patterns that is easy to use and manage both modules and module functionalities. The design of the framework could benefit other developers in extending and managing the functionalities of their own VR exergames. This contribution is included in Chapter 3 and Chapter 4 of the thesis.

The second contribution lies in the evaluation of how the framework benefits the developer in building VR exergames. This framework has been effectively used in empirically developing several VR exergames for user studies and student capstone projects. The framework’s code impact on reducing development effort and increasing development efficiency are quantitatively analysed through case studies that compare the VR exergames built not using
CHAPTER 1. INTRODUCTION

framework with those built using framework. Further, this framework is to be used by other research groups in application development, this infrastructure must fulfil the requirements of real-world application software measures such as extensibility, reusability and maintainability. This evaluation also includes the study on the quality of framework code via code metrics analysis which provides a better insight of the framework code. This contribution is described and discussed in Chapter 5 of this thesis.

The third contribution is the feasibility and usability evaluation of the framework. This framework targets to help the researchers in HCI community to build VR exergames to conduct user studies. Based on this goal, the VR exergames that were previously built using this framework and used for user studies are included in the assessment. The results of how players performed with these VR exergames and how the games engaged players doing physical activities are analysed. This contribution is described and discussed in Chapter 6 of this thesis.

1.5 Structure of the thesis

This thesis is structured as follows: Chapter 2 describes the background through the research literature correlated with the topics of the thesis. It first introduces the technologies used (including hardware devices, software techniques and designs) in virtual reality exergame domain, then existing VR exergame frameworks employed in HCI domain are presented. Chapter 3 describes the methodology used to design the framework. The design principal and framework architecture are introduced. Then the design patterns used, framework data flow design based on modular design and other design details are presented. Chapter 4 conveys the
implementation of detail functionalities in modules and framework, the management
implementation in module and framework, and the object-oriented programming in framework.
Chapter 5 demonstrates the comparison between the VR exergames built not using framework
with those built using this framework through a quantitative analysis evaluation. It also
discusses the software measures used in evaluating the quality of the framework code and the
results of code metrics. Chapter 6 analyses the data from user studies to present the feasibility
and usability of the framework in producing VR exergames for HCI experiments. Chapter 7
concludes the principle outcomes and contributions of this thesis, meanwhile discusses the
suggestions for future work.
Chapter 2. Background

Summary

This chapter presents the literature of research areas related to the design and implementation of this framework. The immersive VR technology, VR exergames and the design essentials are introduced. Then the benefits of using component-based software design, patterns and object-oriented framework for the development of applications are described. The last section provides the backgrounds of existing VR exergame frameworks in terms of partial immersion and full immersion.
2.1 Virtual reality

There are multiple definitions of Virtual Reality for its immersive and interactive. Stuart (1996) describes virtual reality as “systems capable of producing an interactive immersive multisensory 3-D synthetic environment”. According to Sherman and Craig (2003), virtual reality is an interactive medium that mentally immerse the user with computer simulated virtual environments. In the book “Virtual Reality Technology”, Burdea and Coiffet (2003) define virtual reality as “a high-end user-computer interface that involves real-time simulation and interactions through multiple sensorial channels. These sensorial modalities are visual, auditory, tactile, smell and taste.” VR system consists of five components (Figure 2-1), in which the

Figure 2-1. The five classic components of a VR system (reproduced from Burdea & Coiffet, 2003, pp. 13)
system provides immersion for interactive communication through input devices (including three-dimensional position trackers, gesture interfaces, navigation and manipulation interfaces) and output devices (such as graphics displays, sound displays and haptic feedback) (Burdea & Coiffet, 2003).

Though the definitions of VR differ from each other, there are several significant features that contribute to VR: interaction, immersion, presence, engagement, multi-sensory and synthetic.

Immersion is the most significant feature of VR. Slater (2003, pp. 1) defines immersion as “what the technology delivers from an objective point of view. The more that a system delivers displays (in all sensory modalities) and tracking that preserves fidelity in relation to their equivalent real-world sensory modalities, the more that it is ‘immersive’.” Based on different levels of immersion, Muhanna (2015) categorizes virtual reality systems into three classifications: Low immersion stands for hand-based or monitor-based virtual systems which are basic systems; Partial immersion represents large screen projection for virtual scene; Fully immersive systems deliver the virtual 3D environments within a large field of view.

VR systems with low and partial immersion have been broadly used in professional domains ranging from engineering, medicine, education, health to military. For instance, Zhang (2007) presents a mobile VR navigation system which involves Pocket PCs or smart phones as mobile VE computing and display unit, M2S (multi-format, multi-scale and semantic) maps as the technique to transfer 2D maps to real world via 3D environments, to improve the user’s
Spatial cognition in navigation; Hou and Sourina (2011) develop a system that combines haptic device with monitor-based virtual environment enabling the user to manipulate and feel the interactions during molecular docking process (Figure 2-2. Left); Additionally, the W. M. Keck Centre (Keck, 2012) uses a Cave Automated Virtual Environment (CAVE) with three-wall projection to visualize the model of Earth where the user can interact with the texture of Earth (Figure 2-2. Right).

With the advances of technology, fully immersive VR systems that provide better presence in VE are widely adopted in research.

### 2.1.1 Immersive VR

Immersive virtual reality presents a high-fidelity, computer-generated scene together with a number of senses, in which the users could barely be aware that they are inside a virtual environment. The stereoscopic Head-Mounted Display (HMD) is a typical feature of an
Immersive VR system. HMD is commonly used with other body interaction devices. NASA (1990) introduces a virtual interface environment workstation in which the HMD is employed for displaying a 360-degree computer-generated environment and a pair of DataGlove is integrated for tactile interaction with computer scene by manipulating virtual objects. The workstation is also emerged with 3D audio headphone and position tracking sensors for multisensory interaction (Figure 2-3). This system is able to provide a uniform interface for complex operational tasks, e.g. telerobotic or training, meanwhile it is also configurable for human factors research such as multidimensional data visualization and spatial cognition.

However, there are few perceptual issues with early stage of immersion technology. According to Patterson et al.’s study on HMDs (Patterson et al., 2006), low brightness and contrast in HMDs brings perceptual consequences such as eye strain and discomfort; a small field of view from HMD limit the sense of immersion; the tracking inaccuracy and display
update lag could cause a perceptual mismatch between vision and proprioception. All these issues reduce the user’s level of presence in virtual environments (Cakmakci & Rolland, 2006).

In recent years, the advances in input/output hardware, computer performance and tracking techniques make it possible for the immersive VR systems become affordable, high-quality commercial products. Portable mobile VR using head-mounted headset with smartphone as the screen, e.g. Gear VR (Samsung, 2015) and Google Daydream View (Google, 2016), are getting popular in research (Cochrane, 2016; Hock et al., 2017). There are two consumer VR systems, HTC Vive (HTC, 2016) and Oculus Rift (Oculus, 2016), widely used in research with third-party hardware and customised designs, which bring a high presence in VEs with precise interaction with the scene and virtual objects.

As shown in Figure 2-4, both of the systems consist of the following components for fully immersive experience: 1) headset (HMD) with state-of-the-art (immersive 360-degree, high resolution and high refresh rate) display and 3D directional audio to provide visual fidelity, a wide field of view and true-to-life immersive audio; 2) hand controllers with high-definition haptic feedback, multifunction buttons to provide intuitive hand presence in VR; 3) two sensors
to provide a room-scale play area where the head and hand realistic movement and interactions are precisely tracked and translated into VR from the headset and controllers.

With the potential of fully immersive experience in bringing engagement and enjoyment, HCI community has seen an increasing number of research utilizing consumer immersive VR devices, HTC Vive or Oculus Rift, for human factors study through different designs. Following show some examples.

Cheng et al. (2017) integrate a 3D language-learning game with Oculus Rift to explore employing VR technology in teaching cultural interaction. This system is used to simulate an immersive, virtual conversational environment for foreign language, e.g. a learner can learn how and when to bow in Japanese greetings within different contexts through in-game observation (Figure 2-5). Their user study results suggest that immersive VR plays a valuable tool in deeply simulating language-learning environment which improves participants’
involvement and motivation in Japanese culture and physical interaction. Meanwhile participants find enjoyable about the VR experience. However, there are still challenges in such a system: interaction design on user interface adaptation leads to a dizzy feeling for some users; VR technology shows an inconclusive impact on learning itself. According to the authors, future work still needs to be done in increasing the sense of immersion and solving the above issues through a better design.

As shown in Figure 2-6, Zhao et al. (2018) create a simulation system integrated with HTC Vive and wearable haptic controller to help visual impairments navigate in virtual environments, which provides simulation of different feedbacks (physical resistance feedback, vibrotactile feedback and spatial 3D auditory feedback) through the interaction between wearable haptic controller and virtual scenes. This system is used to simulate different indoor and outdoor scenarios with 3D environmental sound and various virtual obstacles (differ in shape, texture and surface). The blind users navigate themselves through recognition of the 3D sound of environment and different feedbacks from different obstacles, by transferring the cane skills from real world into the virtual world. The formative study show that this system is a
promising tool for impaired people to safely and efficiently explore and navigate in virtual spaces, which is capable of building applications for visual impairments to entertain and do mobility training. Nevertheless, there are still limitations in such a VR system: there is no physical resistance in forward-backward direction due to the design with brake mechanism; the simulation of outdoor scenario is limited by the size of physical VR space; hardware is heavy for users to wear.

Furthermore, Goedicke et al. (2018) develop a VR driving simulation that allows participants to experience driving in VE while physically sensing the real-world, through wearing Oculus Rift headset and sitting in a real car. This low-cost simulator can be used as a tool to test the understanding of on-road human response. Boyd et al. (2018) implement an immersive VR social application which uses virtual avatars to visualize and deliver the real-time nonverbal communication cues such as gesture and body language. This application is used for the therapy of autism people and shows positive outcomes. In fact, immersive VR has been broadly used as a basic research tool in professions such as psychology and education (Loomis et al., 1999; Ott & Freina, 2015).

To enhance the immersion, video games are frequently used with VR to promote the engagement and enjoyment. Exercise is also commonly involved with VR as well, to provide physical activity for health in the form of exergames.

### 2.1.2 VR exergame

The role of exergaming (physical activity through video games) in improving physical activity
has been studied by academics. Through their literature review, Peng et al. (2012) point out that active video games are able to offer physical activity on light-to-moderate level. Sween et al. (2014) further add that there is a strong correlation between exergaming and raised energy expenditure. They suggest that a short duration of exergame play (10 to 30 minutes) can reach a medium-vigorous intensity exercise with a significant increase in energy expenditure. The amount of activity in such a duration can reach the health and fitness level recommended by American College of Sports Medicine guidelines, allowing the prevention of obesity. Moreover, playing exergame for a longer duration (1 to 2 hours) can lead to more health benefits. Matallaoui et al. (2017) summarize that elements such as competition, enjoyment, achievement and self-expression in exergames motivate players to do more activities. In addition, the energy consumption in exergame play relies on user’s in-game perception and experience, which can be increased with VR. The combination of exergame and virtual reality to can be a potential strategy to improve health.

HCI community has presented many research focusing on using VR exergames to promote health. Bolton et al. (2014) develop an immersive VR cycling exergame with bicycle, power trainer, Oculus Rift (development kit version) and Kinect (Microsoft, 2012). This set up (Figure 2-7. Left) not only allows players to pedal in the VE without feeling motion sickness through mapping the speed in VE to the speed on power trainer, but also enable players to interact with VE using hand gestures that are detected by Kinect. An in-game avatar is created to increase the sense of immersion. Yet the authors imply that players cannot steer in this system,
meanwhile there are safety issue in using Kinect on a bicycle due to tracking range. Overall, this system brings a high level of immersion which inspire player to do more exercise. There is still future work to be done which focuses on providing better design and better safety consideration.

Another example is an immersive VR exergame for Mild Cognitive Impairments (MCI) to do physical exercise, which is the early stage of dementia (Eisapour et al., 2018). This system consists of several exergames that are played with Oculus Rift, where gameplay follows therapists’ recommendation. One of the games is shown in Figure 2-7 (Right), in which the player rows a boat in VE with both hands waving the VR controllers. Other games involve motions such as head rotation, reaching straight ahead, cross-body reaching and lifting arms. Feedbacks from participants and therapists suggest that the games are enjoyable and engaging as a promising tool for MCI people to do physical activity in a safely seated way. It is also recommended to add more in-game feedbacks and more details in the VE to help participants understand the scenarios, as well as to use better interactive designs.
Besides, immersive VR exergames used in physical functioning improvement (Molina et al., 2014), rehabilitation (Corbetta et al., 2015), balance training (Diest et al., 2013) and social facilitation (Anderson-Hanley et al., 2011), suggest that it is a valuable tool in studies that not only boosts physical, cognitive and psychological health for users, but also promotes better outcomes for researchers.

However, the design, implementation and customisation of such various VR systems usually hinder an HCI group in research, which objective is to conduct human factors study rather than application development.

### 2.1.3 Generalizations of VR exergame design

Though VR exergames and related research vary, we can still conclude the generalizations of the design about it. Consolvo et al. (2006) introduce four design requirements for technologies that encourage physical activity to prevent and decrease overweight and obesity. According to them, employing technologies to encourage physical activity (e.g. exergame) should take four key requirements into consideration: 1) give users proper credit for their activities. 2) provide personal awareness of activity level. 3) support social influence. 4) the practical constraints of users’ lifestyle.

Sinclair et al. (2007) identify success factors in the guidance for designing exergaming systems: attractiveness and effectiveness. The former one represents player’s game experience and the latter one stands for player’s physical condition. An attractive exergaming system should also be effective as an exercise, which makes it a successive level of exergame. It is
also suggested that using principles such as core mechanic, representation, micro goals, marginal challenge, free play, social play and fair play in fitness game design could lead to successful exergames (T. Campbell et al., 2008).

In addition, user feedback, especially visual feedback on enjoyment and perceived competence, plays an important role in motivating physical activity. In the meantime, a list of requirements on integrating feedback into exergames need to be considered. Examples are: application should be general; interaction complexity should be low; feedback components should be flexible to add or remove; feedback should respond to user needs, etc. (Doyle et al., 2011).

Planinc et al. (2013) further presents design guidelines to develop exergames for elderly, which foster them to stay active and improve their overall wellbeing, as well as enhancing their mobility. Parts of the guidelines are: use appropriate gestures; avoid small objects; give visual and auditory feedback; adjust the difficulty; use a clear user interface; use a suitable topic, etc.

It is similarly concluded that utilize physical space and sensor technology allowing for multi-direction movements is important, meanwhile providing the representation of movements. Another important element is the natural mapping in the game narrative, which benefits movement quality and game outcomes (Skjær et al., 2015).

When it comes to VR exergame, the design consideration includes not only exergame but also virtual reality. In the VR exergame design, several major significant design aspects are summarized by Shaw et al. (2015). First, VR exergames should try to avoid motion sickness
brought by sensory disconnect when using HMD. Mapping the position, movement or speed in VE to the activity in real world can efficiently reduce motion sickness. Second, VR exergames should effectively provide reliable bodily motion tracking controls in terms of high accuracy, consistency and suitability. Third, VR exergames should be healthy and safe for users. A warm-up stage in exergames is necessary to be included to lower the risk of getting injured during exercise. User safety is very important while using HMD because players cannot see the real world and the cables attached to the VR devices. Another safety requirement in design is to prevent players from losing balance, which may lead to falling off from exercycle while wearing HMD. Fourth, VR exergames should choose an appropriate player perspective. The selection of a proper perspective helps players to easily understand the view system. Last, VR exergames should have low physical feedback latency, which means the associated sensory feedback from the game should respond immediately when the event is triggered in game.

In summary, a proper VR exergame design should meet the requirements and principles identified above, which then brings the possibility of conducting successful user study.

2.1.4 Game engine for VR development

Since virtual reality gets popular in both entertainment and academic, the number of tools available for developing modern VR applications is also increasing. The most frequently involved tools for VR development are two contemporary game engines: Unity (Unity Technologies, 2006) (Figure 2-8) and Unreal Engine 4 (Epic Games, 2014). These two modern game engines provide simple user interfaces to build three-dimensional game scenes and cross-
platform, real-time simulations. A set of functionalities are also offered to support libraries and third-party plug-ins, e.g. the support for VR devices. The two engines use a same popular entity-component-system architecture pattern to provide a great flexibility in game development (Nystrom, 2014). In this architecture, every object in a game scene is an entity, for example, each vehicle, light, tree, or audio source, is a single entity. An entity can be seen as a container, which contains different basic script components. In this way, an entity’s features, such as appearance (colour, shape, etc.) and behaviour (rotate, move, position, etc.), are provided by the component scripts added to this entity. The entities group together different components, where each component contains data and basic behaviour. Therefore, the functionality of an entity can be changed at runtime through adding or removing component scripts. Both Unity and Unreal Engine use this architectural pattern and similar user interface
for development. HCI community generally employ Unity for VR application development, based on the fact that 47% of the applications developers are using Unity and 45% of global game engine market is possessed by Unity, according to Unity’s website (Unity Technologies, 2016).

Figure 2-8 shows a screenshot of Unity Editor (version 5.6.0f3) with its user interface, there are several main windows in this interface. The Scene Window (Figure 2-9. Left) shows the interactive view of game scene, where developer can add and position different entities (in Unity, entity is GameObject, which is the basic object in a scene) through this interface. The Game Window (Figure 2-9. Right) represents the final rendered game, which is the view from player’s perspective. The Hierarchy Window (Figure 2-10. Left) comprises a series of GameObjects in the present scene, where GameObjects can be set as parent object or child object of each other. The Project Window (Figure 2-10. Right) displays the structure of the folder in this project and the contents in the chosen folder. The Inspector Window (Figure 2-11) presents all the component scripts, variables and properties contained in the selected
CHAPTER 2. BACKGROUND

GameObject, where the functionality of this GameObject can be modified.

To build a VR project in Unity, the developer firstly needs to import the VR Software Development Kit (SDK) or plug-ins into Unity through Project Window, which contains all the Application Programming Interfaces (APIs) to use the hardware and the 3D models (in the form of GameObjects) of them, usually headset and hand controllers. Secondly, the GameObjects of VR headset and hand controllers need to be added into the scene by dragging them from Project Window and dropping into Hierarchy Window, meanwhile the position can be set through Scene Window. Thirdly, the default camera in Unity needs to be disabled and the VR camera in VR headset GameObject needs to be enabled via Inspector Window. Then the developer can run the real-time simulation of VR scene through Unity Editor, where the player’s view will be shown in Game Window. Besides, the developer can adjust variables and properties in Inspector Window and write code using Unity’s script editor MonoDevelop-Unity or Microsoft Visual Studio (Microsoft, 2015b), then run the test simulation in Game Window.

Figure 2-10. Hierarchy Window (left) and Project Window (right) in Unity interface, sourced from https://docs.unity3d.com
The above description is the regular way of building a VR application through Unity, which only includes adding and enabling VR in Unity. Other objects in virtual scene and the interaction between VR hand controllers and objects still need to be customised by developer, which is the main part of VR application development and costs most of the time. However, if similar VR applications are going to be built, especially in which there are many repetitive code and functionalities between projects, exploiting the applications from scratch every time will be a time-consuming job that hinders the research. In software engineering a typical way to solve the issues in code reuse is to employ the framework for software development.

Figure 2-11. Inspector Window in Unity interface, sourced from https://docs.unity3d.com
2.2 Software framework

As defined by Roberts & Johnson (1996), frameworks are “reusable designs of all or part of a software system described by a set of abstract classes and the way instances of those classes collaborate”. It is created to reduce the effort in software development through reusing the design and code. Framework is also introduced as a reusable structure that consists of abstract classes for solutions to similar problems, which is able to be specialized for customising applications (Fayad et al., 1997; Johnson & Foote, 1988). A software framework provides a general method or standard functionality to develop applications, which can also be extended and changed based on user’s code. Fundamentally, framework can be seen as a skeleton of a software application in structure, and its purpose is to be customised by developer for different applications (Johnson, 1997).

As an object-oriented reuse technique, framework differs from reusable components and design patterns. Johnson (1997) argues that on one hand framework is more than software component because it can be customised and it has more complex interfaces. On the other hand, framework is a form of design reuse but it is expressed in code, which means framework is more than ideas or patterns. He also suggests that framework has more abstraction and flexibility than software component, as well as that framework is more concrete and applicable in reusability than design patterns. In summary, framework is a program or a template to generate applications that reuses code and design. However, it is also pointed out that framework is usually hard to develop, and it requires time to learn from the documentation or
training. Nevertheless, framework is still a valuable tool in software development which brings the advantages from both components and design patterns.

Early frameworks are developed by companies, Microsoft (Brockschmidt, 1995) introduces the Object Linking and Embedding (OLE) technology to enable application programs to share compound documents from different sources. OLE defines a mechanism and a protocol, allowing application programs to link and embed to reusable documents and other objects, which is called Component Object Model (COM). To be specific, OLE creates a master file and a compound file binary format document referring to the master file. Different programs can make changes to the data in master file, meanwhile it affects the referred document at the same time. In this way, different application programs that process text, sound, image, table, etc., can be combined to edit one file. This framework significantly benefits developers using Microsoft Windows interface system.

Apple Inc. (1995) announces the platform-independent software componentry framework, named OpenDoc, for the development of component-based application programs. OpenDoc framework provides reusable components accomplishing different task (e.g. sound processing and image editing), and a shared document storing the data generated by every component. In this way, this document centred framework helps the developer to focus on the development of a single part, leaving the end-users to choose which other parts of functionalities to be mixed and matched together.

Sun Microsystems Inc. (Hamilton, 1997) releases JavaBeans framework for the reusable
software components of Java. It is a set of classes (APIs) encapsulating diverse objects into one object (the Bean). This framework uses Java Object Serialization API, which brings the advantages of serializability and reusability. In addition, a Bean can determine whether to expose its properties, events and methods to other applications or not. Further, a Bean can receive events from other objects and generate events that are delivered to other objects. Meanwhile a Bean is also configurable through software.

From the early examples of software framework, we can see that in a proper framework, there are two main ingredients: reusable components and specific design patterns.

2.2.1 Component-based software engineering

In computer science, Component-Based Software Engineering (CBSE) is a technology that focuses on the separation of concerns, which is a design principle that divides a computer program into different components, in regard of the functionalities in a software system. Clements (1995) introduces that software engineering for components dividing the software architecture into reusable and interconnected components brings new potential advantages in component-based software engineering. It reduces development time, because it takes much less time to use a component from a supplier, which can be reused in different systems, than building the component. It also increases the reliability of a system and flexibility of components. Additionally, component-based software engineering unifies concepts from software domains such as object-oriented programming and software architecture. Several component infrastructure techniques have become standardized: Microsoft’s Component

Component-based software engineering has brought the new way of software development from monolithic architecture and conventional “analysis-design-implement-test” process to a modular architecture and composing process with reusable “plug & play” software components (Aoyama, 1998). Cai et al. (2000) present this promising development approach in Figure 2-12. In this approach, software development no longer starts from scratch. Instead, different components that developed with different languages and platforms can be selected from a component repository and assembled into a new software system. The research community and industry practice (Griss, 1997) have concluded the architecture of component-based software systems, which is layered and modular (Figure 2-13). Each component layer contains different components that are reusable in diverse applications. Though a system can be made up of numerous components deriving from various sources, the process of selecting proper
components for system is of great importance (Crnkovic, 2001), especially the identification and selection of usable and suitable components. The design patterns community provides general components as a solution.

2.2.2 Design patterns

The idea of pattern was first introduced as an architectural concept (Alexander et al., 1977), which was then applied in programming and proved to be successful in user interface design (Beck & Cunningham, 1987). Generally, a pattern represents a universal solution to the same type of development problems within a particular context. In the context of software design that certain pattern applies, the pattern captures replicated designs and strategies for solving the recurring problems, so pattern benefits developers with best practices in development (Gamma et al., 1994; D. Riehle & Züllighoven, 1996). Since associated patterns can form a pattern language (Gabriel, 1994), which is not a formal language but a structured collection of
interrelated patterns to guide the user towards valuable architectures, both patterns and pattern languages help developers to communicate software design experience (Schmidt et al., 1996).

Design patterns get widely used in computer science because of Gamma et al.’s book (Gamma et al., 1994), which discusses the capabilities and pitfalls in programming, then introduces 23 classic patterns in software design and related examples. Successful patterns employed in common architectures for communication provide preserved design information to build flexible and efficient software, for example Reactors (Schmidt, 1995), Active Objects (Lavender & Schmidt, 1995) and Brokers (Buschmann et al., 1996).

As stated by Coplien (1998), a pattern describes part of a product and the process to build this product. Meanwhile a pattern differs from a paradigm, a rule or an idiom, because a pattern describes different parts and the relationships between them, as well as it needs to be interpreted by architects to be applied within the context as an architectural technique. On the other side, Brown et al. (1998) notice that patterns can also encapsulate bad practices and provide the approaches to deal with them.

Though successful patterns provide abstractions and reusable designs to benefit developers in reducing effort and maintaining cost, they do not yield reusable code. Therefore, it is essential to combine design patterns with reusable components for creating object-oriented applications, which contributes to the framework (Schmidt, 1997).

### 2.2.3 Object-oriented application framework

Framework uses object-oriented programming language to benefit developers in its modularity,
CHAPTER 2. BACKGROUND

reusability, extensibility and inversion of control (Fayad & Schmidt, 1997). Through encapsulating the details of implementation behind user interfaces, framework is a modular design with specific code. It is simple to use since developer does not need to understand the implementation detail. The shared components defined in a framework can be reapplied in developing new applications, which brings the advantage of reusability. In addition, framework provides hook methods, which decouple the interfaces and implementation to allow the extensibility in interfaces (Pree, 1995). Furthermore, framework, instead of custom code, can invoke the task-specific code, allowing the inversion of control.

Object-oriented application frameworks have been widely applied in certain areas, where a typical domain is Graphical User Interface (GUI). An early example, MacApp (Apple Inc., 1985), is Apple Computer’s successful GUI framework in creating GUls for classic Mac OS. Another GUI framework that gains widespread adoption is Microsoft Foundation Classes (MFC) (Microsoft, 1992), which has been a contemporary standard for building graphical applications on windows system. The reuse of common frameworks in graphical application development has improved the productivity in industry.

Application frameworks can be categorized by scopes into system infrastructure framework (R. H. Campbell & Islam, 1992; Schmidt, 1997), enterprise application framework (Hamu & Fayad, 1998) and middleware integration framework (Otte et al., 1995). They can also be categorized by technologies employed such as white box framework and black box framework, e.g. framework uses Template Method pattern (white box) and framework uses strategy pattern
Object-oriented frameworks significantly decrease development effort and improve software quality through design patterns, reusable components and class libraries. Nevertheless, there are still challenges in the process of building and using frameworks (Fayad et al., 1997). Developing a proper framework with high quality, reusability and extensibility requires a wide range of knowledges and skills, which can be much more difficult than developing a complex software. The maintenance of a framework can also be a tough work including modifying, extending and adding functionalities. Further, it usually takes long time to efficiently and productively master a framework like MFC. Although there are industry standard frameworks such as Java Remote Method Invocation (RMI) (Oracle Corporation, 1997) and Common Object Request Broker Architecture (CORBA) (Object Management Group, 1995), it is lack of standard in the design, implementation and documentation of frameworks.

Given the challenges in frameworks, it is still of great importance to create frameworks that uses design patterns, reusable components, software architectures, middleware modules or protocols to reduce the complexity in software development and customisation.

### 2.3 Frameworks for VR exergames

Virtual reality has been used as a tool by HCI community for many studies, which contributes to a growing number of VR frameworks built by researchers and expert programmers. Frameworks such as SSVE (Linebarger et al., 2003), basho (Hinkenjann & Mannuβ, 2004), Avango (Kuck et al., 2008) and SOFA (Allard et al., 2007) employ white box designs and
object-oriented programming language C++ to provide reusable elements for building dynamic and collaborative applications, distributed applications and simulation applications with virtual reality. However, they include neither certain design patterns nor reusable components.

There are also VR frameworks that use both design patterns and components. For example, Kapolka et al. (2002) build an open source VR framework with Model-View-Controller (MVC) pattern and component-based architecture. This framework utilizes existing standards to ease the configuration in building modular applications, and interface layer for extensibility in networked virtual environments. Maciel et al. (2011) introduce a software framework to develop collaborative virtual environments. With the use of MVC pattern, their framework supports the development of multimodal VR applications for collaborative surgical simulation involving haptic devices. Elvezio et al. (2016) present a framework to facilitate creating real-time interactive systems in Unity. Following an object-oriented pattern, their component-based framework is well designed in flexibility and extensibility to enable a rapid development of a modular system with virtual reality.

Even though there are quite a few VR frameworks published to facilitate the development of VR applications, frameworks that focus on supporting exercise with virtual reality are scarce. This thesis presents the literatures on frameworks for developing VR exergames in terms of partial immersion that uses monitors or large screen projections, and full immersion with latest fully immersive VR devices.
2.3.1 Partially immersive frameworks

- **XVR framework (Carrozzino et al., 2005)**

Carrozzino et al. (2005) introduce a framework named XVR for the development of multimodal interactive applications. Integrated with Web3D (Walsh & Bourges-Sévenier, 2000), an interactive 3D technique via web browser, and a number of interactive devices, XVR is a development environment that proposes a general architecture for VR applications.

One side, XVR provides the support for different VR devices including real-time motion trackers, 3D mice, haptic interfaces, sensorised gloves and stereo projection systems. Together with 3D graphics engine, this setup enables the content creation for complex VR experiences involving 3D animation, spatial audio effect and user interactions.

On the other side, the XVR platform is embedded with various tools to ease the design and development of VR prototypes. It provides an XVR Studio (Figure 2-14) Integrated Development Environment (IDE), which contains different windows for editing scripts, managing workspace, compiling source code, publishing project with FTP and tracing debug messages. This studio supports an XVR scripting language (S3D) to manage 3D contents, which wraps Open Graphics Library (OpenGL) functions into high-level functionalities.

The modular design of XVR offers extensibility for additional modules with advanced functionalities, e.g. motion trackers and haptic gloves. The flexibility of XVR framework is illustrated through the development of different motion-based VR simulators, as well as the interactive projects developed by university students.
In summary, this framework provides the integration of different devices and the development of VR contents. Though code reuse is achieved through the functionalities within scripting language, it is not specified of using any design patterns in software layer or structure as the scripting language simply wraps OpenGL classes.

With the support for multimodal interactive devices, the XVR framework can be used to customise different VR exergames. For instance, Ruffaldi et al. (2011) build a training system using XVR framework, which simplifies the process of devices integration and real-time visualization. As shown in Figure 2-15, the SPRINT system consists of a pair of fans, a rowing system, large screen display and sensors. This system uses VE and multimodal feedbacks (both visual and vibrotactile) from sensors for rowing exercise and sensorimotor training.
• **PhysX-based framework (D’Andrea et al., 2013)**

Since virtual reality benefits the outcomes in rehabilitation and therapy, researchers begin to build frameworks for developing VR rehabilitation applications. D’Andrea et al. (2013) present a framework for using haptic interactions in monitor based virtual reality. As illustrated in Figure 2-16, this framework provides the support of diverse visualization interfaces and haptic devices for customising different exercises, where Extensible Markup Language (XML) is used to enable an easy configuration. Based on the MVC pattern, different functionality components (interaction, simulation and visualization) are separated from each other, allowing an independent structure in which the interaction between components are through data. Besides, C++ programming language with software patterns (Strategy, Observer) and PhysX engine (NVIDIA Corporation, 2008) are adopted for real-time simulation of the VE. Basically, this
framework can be customised to produce various applications according to the types of rehabilitation with virtual reality and haptic devices.

It is demonstrated that this framework can save time and cost through the development of a portable and highly configurable prototype in rehabilitation (D’Andrea et al., 2015). Generally, this framework is flexible and adaptable in building VR applications for finger or hand exercise rehabilitation, since devices can be changed according to patient disability and target therapy, meanwhile quantitative information can be recorded during rehabilitation sessions. Though this framework only supports VE in monitor display, which limits the immersion and usage scenario, it is still of great significance in hand rehabilitation.

- **Framework by Avola et al. (2013)**

To assisting the rehabilitation after stroke, surgery or other residual impairment situations, Avola et al. (2013) describe an open source framework for fast prototyping customised VR exercise applications, free to use in research and education. With the goal of helping therapist
to easily implement and modify functional recovery exercises regarding specific patients, this framework combines vision-based interfaces with VE to enable real-time gesture recognition and 3D rendering for rehabilitation evaluation.

According to Figure 2-17, this modular framework consists of two layers. The platform layer includes operating system and Compiler Module, which uses C++ compiler to support the open source real-time 3D engine, Irrlicht Engine (Gebhardt et al., 2006), and C# compiler to support the gesture middleware for 3D camera, iisu SDK (Sony Depthsensing Solutions, 2008). The Business & Presentation layer acts as the processing core which includes three parts: API camera components for acquiring spatial information with 3D camera; Library Module component for enabling data communication and 3D scenario rendering; Development Module for integrating the user avatar with VE and extending the libraries (e.g. defining the behavioral rules and interaction rules). In conclusion, this framework supports the development of VR applications that captures real-time body movements and renders corresponding avatar positions in virtual space containing movements information for task analysis.

With this framework, developers or engineers can customise 2D screen-based VR
applications according to therapist’s requirements through defining the exercise database in regard to multiple rehabilitation purposes. Furthermore, therapists can also prototype new exercises via modifying the XML file stored in web-based server, which improves the evaluation of different tasks and reduces the development time. To summarize, the proposed framework is proved to be an efficient tool that substantially saves cost and time in building VR applications for rehabilitation exercises, through several pilot examples. However, one limitation of the framework is that the open source 3D engine does not support the VR technology with HMD. Another limitation is that none of any design patterns is used to ease the programming process.

- **Framework by Camporesi & Kallmann (2013)**

Collaborative and immersive VR is getting popular in domains such as motion modelling, physical therapy and interactive motion visualization. Pursuing full-body interaction with virtual avatars, Camporesi & Kallmann (2013) create a framework for producing immersive collaborative VR applications, which targets to improve the rendering performance in graphics engines and lack of high-level programming tools in development.

Following a similar Master-Slave architecture introduced in VR toolkits Vrui (Kreylos, 2008) and Avango (Kuck et al., 2008), this framework uses a core to handle other modules in system. According to the system architecture in Figure 2-18, the system core includes a Device Manager to allocate interaction devices, a Camera Frame to manipulate camera and create render array, and a 3D GUI Manager to handle user interaction. Outside of the system core, the
Character Animation module calculates and visualizes the tracked motions, and the Motion Reconstruction module reconstructs the full-body tracking from camera sensors or finger motion from data-gloves into virtual characters. Fundamentally, this system can handle different devices such as 3D cameras and 3D mice. Motion data from cameras and interaction data from other devices can be reconstructed, while the OGRE engine (The Ogre3D Team, 2005) renders the reconstructed data and movements regarding real-time full-body tracking into avatars that can be displayed as VE through multi projection systems. With the support of network communication, two users can see each other’s virtual character with real-time simulated movements of them.

With this framework, developers can employ a number of applications with full-body interfaces and VE. For instance, an interactive training application can be built, in which the users can follow the reproduced avatar motion from an expert to accomplish parameterized motions, through a large screen projected VE. Another important scenario can be physical rehabilitation, where the therapist can directly teach and monitor the patient’s full-body therapy.
exercise with the virtual characters. This framework’s features in scalability, customisability and cross-platform have been demonstrated in different physical configurations by authors, which indicate it a good tool in developing collaborative VR applications especially VE with wall projections. Yet, its source code only provides library classes, without using any reusable structure design or patterns to simplify the programming.

- **Framework by Kikuchi (2014)**

Aiming at providing a safe and human-friendly environment for elderly to do aerobic exercises such as walking and cycling, Kikuchi (2014) proposes a framework for rapidly constructing new contents for VE. Containing open source digital map OpenStreetMap (OpenStreetMap Community, 2004), commercial software CityEngine (Esri Inc, 2008), 3dsMax (Autodesk Inc, 1996), Metasequoia (Tetraface Inc, 2012) and TexturePacker (CodeAndWeb GmbH, 2010), their framework uses DirectX-based software to generate large-scale data that can be rendered in virtual scene.

Figure 2-19 (Left) shows the overview of this framework. When building new contents for VE, the framework firstly provides the functionality of exporting the digital and editable map data from the free world map OpenStreetMap. Then the data is imported to 3D modeling software CityEngine, which converts it to a 3D virtual city with static objects of buildings. Next, the remodeled data is imported to 3dsMax for modifying the objects and textures, where the TexturePacker software is used to bridge texture data and original data. Finally, the data is sent to 3D polygon modeling software Metasequoia, in which moving objects are added. All
the finalized map data, together with the configuration file including trajectory, sound and force, are rendered into an interactive virtual scenario.

Figure 2-19 (Right) presents a setup for psychophysics experiment built with this framework (Kikuchi & Tsubata, 2012). This system projects the interactive VE of a city to a large screen. A user can use the bike to ride in the virtual scene, where the scene moves according to cycling speed and bike’s heading direction. At the meantime the sound simulation of surrounding cars and pedestrians are played through a speaker. This system is designed to understand user’s reaction towards traffic environment when cycling with a cell phone.

This framework focuses on providing an easy tool to develop an interactive VR system for elderly people to do aerobic exercise. With the 3D modeling of the real-world map, support for
ambient environment simulation involving car trajectory, sound and haptic information, as well as the clear structure of data flow, this framework enables a simple and rapid method to construct an interactive virtual city with a large scale. Whereas the VE produced with the framework can simulate an unsafe traffic scene without any real dangers, which could lead a promising aerobic exercise to prevent dementia, the framework only gives instructions of creating computer graphic data step by step rather than providing a whole software architecture for developing new applications. In addition, the cycling method employed can be replaced with a safer design for seniors.

2.3.2 Fully immersive frameworks

- **RUIS framework (Takala, 2014)**

To help students create applications in an annually VR course, Takala (2014) introduces a VR toolkit named RUIS with spatial user interfaces. The Reality-based User Interface System (RUIS) is designed and implemented to resolve the difficulties in immersive VR development regarding 3D user interfaces.

There are several challenges that student face in the development of immersive VR applications. It requires many tasks and skills in computer game development, employing 3D projection or motion tracked controllers, design and development of 3D user interface, etc. RUIS is created to solve the above challenges, with the features that: it simplifies the VR application development process; it can be used with normal PC; it supports simulating motion
trackers with regular input (mouse or keyboard); it provides 3D manipulation utilities; it is free to use and so on.

The RUIS framework is based on Unity game engine, which facilitates the game development process with its easy-to-use interface and editor. Unity has free version for personal use and it supports different input and output devices. Figure 2-20 presents the overview of RUIS software architecture. It has an InputManager to deal with the signals from different input devices and a DisplayManager to for outputting the VE. The core of this framework is 3D User Interface Building Blocks, which is a spatial user interface supporting 6-degrees-of-freedom hand controllers involving PS Move controllers (Sony Interactive Entertainment, 2010) and Razer Hydra (Razer Inc, 2011), Oculus Rift HMD and Microsoft

Figure 2-20. Overview of RUIS architecture (Takala, 2014)
Kinect. Additionally, this framework core supports the use of different movement capturing devices in same coordinate system by a calibration step. With this framework core, a developer can directly add the support for devices through a drag and drop operation in Unity Editor, without manipulating scripts. In this way, developing a VR game with RUIS is the same as developing a normal game with Unity.

Since different devices are included for a full-body interaction in this framework, it can be utilized to build different immersive VR exergames. Figure 2-21 shows two examples. The Left one is a sandbox demo of a virtual world, where the user interacts with the scenario through Kinect full-body tracking and Oculus Rift HMD. The virtual character follows the user’s action to do all the exercises in exploring the virtual scene. The right one is a wheelchair exergame, where a player rotates the wheelchairs’ wheels (wheels are lifted from the ground) to control the wheelchaired avatar to roam a virtual town. Stereo 3D projections are used to present audience the game scene.

To summarize, the combining of Unity with spatial interaction devices of RUIS framework...
makes it capable in developing spatial interfaces for different immersive VR exergames. It also allows a simple development, because the game locomotion is integrated with player’s motion in full-body tracking. However, this framework doesn’t reveal any patterns involved to ease the programming process or enhance the extensibility.

• Framework by Paraskevopoulos et al. (2016)

Though VR technology is widely used in therapy with exercises, the framework for building immersive VR rehabilitation applications is rare. Due to the capability in promoting health and wellbeing, serious games have been used as tools for research for a long time. There is a need in developing serious games with state-of-the-art technologies, for example virtual reality.

To provide a tool for researchers to customise home based VR serious games, Paraskevopoulos et al. (2016) develop a framework for producing holistic VR games for motor rehabilitation. This framework aims at introducing a new idea on VR prescription of therapy exercises in a disease agnostic manner. By mapping traditional therapy to equivalent game mechanics, this framework concludes to a conceptual design that can translate the rehabilitation into fully customised exercise prescription with VR.

As depicted in Figure 2-22, the framework has a three-layer architecture to separate the development process of an application. The Input Layer is where developers add input hardware devices to system. It supports various devices, including VR headset, Kinect, Balance Board (Nintendo, 2007), etc., to provide motion capturing and mapping to interact with objects. In this separation method, the rehabilitation intervention will not be affected by the rapid
improvements in technology. On one hand, the motion data from input devices are sent to Game Layer through WiFi or bluetooth as the interaction data. On the other hand, the motion data are transported to Clinician Layer for further calculation and display (Paraskevopoulos & Tsekleves, 2013). The Game Layer is where the game contents are created. On this layer, developer can create game scene which supports input calibration and automatic difficulty adjustment. This layer provides the functionalities of motion mapping for objects interaction and game feedback for monitoring. It is also connected to Clinician Layer in sending game feedback and receiving clinician input. The Clinician Layer allows developing the mapping of exercises to game mechanics and customising the VR physiotherapy prescription. On the other side, it also

Figure 2-22. Schematic diagram of the proposed framework, reproduced from Paraskevopoulos et al. (2016)
visualizes the real-time movement of a user, as well as both recorded and real-time biofeedback. The visualization of movement and data enables a performance monitoring of a remote user. In particular, live communication is supported in this layer, which facilitates the remote intervention.

With this conceptual framework, developer can create immersive VR exergames for rehabilitation which not only have the advantages in customising remotely co-created and playful game contents with the involvement of service users (through translating exercise motion to game control) and physiotherapist (via prescribing the parameters of repeated exercises), but also enables therapist a remote real-time monitoring of user performance and live communication for intervention. This framework has been demonstrated as a feasible tool with great customisability in developing therapy exercise sessions in pilot studies. The separation of layered architecture leads to efficacy in developing process, yet this framework doesn’t provide any solutions to simplify the virtual scene construction and programming.

- **Frameworks by Shepherd et al. (2018)**

  In sports domain, video playback is a traditional way for performance analysis. However, techniques in hardware and software of video playback have limited the understanding of athlete’s perception-action loop, for example, the fixed camera viewpoint and essential specific video recordings. To overcome these limitations, Bideau et al. (2010) design a system that uses Vicon system (Oxford Metrics Group, 2004) to capture sports movements motion, MKM engine (Multon et al., 2008) to animate sports actions and Cybermind HMD (Cinoptics, 2005)
CHAPTER 2. BACKGROUND

Figure 2-23. Handball goalkeeper using the system to improve performance (Bideau et al., 2010)

to display the interactive virtual environment. This system enables presenting the simulation of an athlete’s captured motion in immersive HMD that is used for analysis on both perception task and perception-action task to improve the performance of handball goalkeeper (Figure 2-23).

Arguing that Bideau et al.’s system still has limitations when employed in other sports, Shepherd et al. (2018) propose an operational framework with more generalizable guideline, based on Bideau et al.’s system concept, for designing training simulator with virtual reality, which enables ordinary users to experience unattainable sports performance situation, as well as assisting elite athletic training for a better performance in sports competition.

Illustrated in Figure 2-24 (Left), this technological design framework includes adapted criteria in enhancing perpetual fidelity and representing the behaviours (Miles et al., 2012), comprised of the following four steps: 1) recording reference data; 2) developing software; 3)
integrating hardware and 4) analysing and iterating. Steps are explained as follows during the development process of a track cycling simulator.

The reference data is initially recorded from an elite cyclist with three inertial sensors (SABEL Labs, 2009), a 360-degree camera (360fly Inc, 2015), a fixed sports action camera (GoPro Inc, 2015), a GPS sports watch (Polar Electro, 2014b), a heart rate monitor (Polar Electro, 2014a) and a bike computer (Garmin Ltd, 2015), as shown in Figure 2-25 (Left). Then the reference video data are used to produce high-fidelity models through 3dsMax and Photoshop, as well as animated within Unity in regard of data from sensor. The VE rendered by Unity can be displayed in Oculus Rift HMD. Thirdly, in the implementation of simulator, a stationary exercise bike (Wattbike Ltd, 2016) is connected to a PC for sending real-time performance data. Depicted in Figure 2-25 (Right), the simulation experience involves the virtual scenario of a track and a virtual opponent competitor riding against the user. The speed of avatar is generated based on reference data and it will change depending on user’s speed.
Final step is user testing, where non-elite riders compete with the virtual riding character of elite cyclist without knowing that opponent’s speed changes according to theirs. During testing, all the cycling related data can be recorded for performance analysis.

This technological design framework provides a four-step approach and component system to guide the creation of a sports simulator. The feasibility of this framework is demonstrated through building a track cycling simulator and the user testing, which allows users to comprehend the reason of their behavioural change and leads to an actionable insight. On one hand, the proposed hardware setup of the framework enables the creation of a high visual fidelity simulator for different sports. On the other hand, the user testing in track cycling simulation confirms the significance of using immersive VR technique in performance measurement and improvement. This framework focuses on offering a guidance for designing training-based VR simulator for sports performance analysis, meanwhile a framework architecture for track cycling is presented, however, it doesn’t reveal any reusable software
components in implementation especially programming stage.

- **Framework by Semblantes et al. (2018)**

Given that the traditional treatment in stroke tend to be boring, exhausting or expensive for patients, researchers have been exploring new technologies in treatments to increase the enjoyment and get better rehabilitation outcome, where virtual reality is one of the broadly used solutions with best performance (Szmeková et al., 2015). Nevertheless, different advances need to be developed when using new technique to validate rehabilitation exercise in different areas, for instance, various alignment algorithms and classifiers (Petitjean et al., 2016). Thus, Semblantes et al. (2018) present a framework for building exergames for stroke rehabilitation with visual feedback.

Figure 2-26 (Left) shows the structure of a stroke rehabilitation system involving new technologies, virtual reality and motion sensing device. In this system, patient uses Oculus HMD and Kinect v2 (Microsoft, 2014) to interact with the virtual scene. Therapist validates the data captured by Kinect v2 and then evaluates treatment progress, sets new exercise routine.
Semblantes et al. design the framework with the same structure, illustrated in Figure 2-26 (Right), where Unity is used as the game engine and MATLAB is used for analysis.

This framework’s architecture is composed of five component blocks (Figure 2-27): 1) the system inputs block includes two devices. Oculus Rift HMD provides the functionalities of displaying immersive virtual scene and head orientation tracking, encouraging the exercise performance. Kinect v2 supports the position tracking of human body’s main joints for accurate measurement. 2) the algorithm design block uses algorithms, e.g. Dynamic Time Warping (DTW) algorithm (Kyan et al., 2015), to calculate the alignment between two vectors that represent the human body limbs, one from user during exercise routine and one from the pre-
defined model by therapist. 3) the communication block handles the data transmission between Unity and MATLAB with Comma Separated Values (CSV) file, which enables data recording, reading and analysis. 4) the VE block, running on Unity engine, has an intuitive interface showing the virtual scene and virtual objects to motivate patient to accomplish exercise sessions. All the virtual scene and virtual objects are pre-designed in Unity and game mechanics are pre-written scripts in Microsoft Visual Studio. 5) the outputs block emits sound and visual feedback within HMD to enhance the immerssion. It also produces the result of task execution for therapist to evaluate.

The usability and simplicity of this framework has been demonstrated in the development of a rehabilitation system with a series of VR exergames. The immersion of exergames developed and the positive influence on exercise tasks have also been proved through experiments, in which the exergames hold users’ attention and keep them continuing doing tasks. In conclusion, with the help of this framework, developer or therapist can easily produce exergames for stroke rehabilitation with Oculus Rift HMD and Kinect v2 that are suitable for home setup. Virtual reality increases the motivation for patient exercise and motion sensing hardware allows a precise performance evaluation by specialist. However, a new motion sensing device needs to be adopted when using this framework, because Microsoft has stopped producing Kinect since 2017. In addition, this framework doesn’t include any pattern to ease the programming process in building virtual scene, adding new devices or new game mechanics.

The literature review in this chapter underlines the scarcity of work creating software
framework to produce VR exergames that integrated with different devices. Given the challenges that building such a framework is not only a time consuming and effort costing mission, it also requires a lot of domain knowledge in design and programming. A proper software framework benefits the future development of applications in many aspects for both the research group and HCI community. Additionally, the literature reveals the significance of building a framework for immersive VR exergames, but it still lacks object-oriented frameworks that provide reusable software components and design patterns in HCI domain.
Chapter 3. Framework Infrastructure

Summary

After building several VR exergames for user studies from scratch, we recognised the significance of using software framework to simplify the development process of future VR applications. Therefore, an object-oriented framework is designed and implemented. This chapter introduces how the domain knowledge are abstracted into design patterns, as well as the architecture of framework. Then the details of our module design are presented with the hardware involved and design patterns used. The framework data flow is also introduced. This chapter extends and takes partial description from the two published papers (Wang et al., 2017, 2020) listed in the Preface section, where I am the lead author.
3.1 Introduction

The literature in Chapter 2 yielded that the successful software frameworks for VR application development in HCI focused on providing different reusable software components but lacked the involvement of design patterns that abstract the domain specific requirements. Though each framework integrated different hardware devices to support safe exercise and exercise feedback, none of the frameworks provided a VR scene that is explorable based on geographic locations for simulating real-life exercise experiences. Besides, some essential features for creating experiment-oriented immersive VR exergames are still not available. For example, features such as the reusable design that are safe and suitable for young people to do moderate exercise, and elderly to do low-intensity physical activity; as well as the design for developing applications that can handle the variables and parameters for grouping in HCI user study.

Through the development of multiple VR exergames for different HCI research projects, the domain specific knowledge could be abstracted into design patterns for reducing effort and improving the quality of applications in new application implementations. In this way, the knowledge from previous experience is reused through the patterns, e.g. reuse of structures, interfaces and designs. Since OOAF is the most effective approach for incorporating the domain specific knowledge into designs, an OOAF meeting the domain requirements for creating experiment-oriented immersive VR exergames shows significance in usage for HCI research groups.

In the domain of HCI, many user studies require the development of software applications.
Over the last few years, our research group have developed several VR exergames (Ijaz et al., 2016b, 2016a) using the Unity game engine, VR platform and C# programming language.

Our early VR exergames employ a same setup (Figure 3-1): a stationary trike (GreenSpeed, 2012) for easy and safe cycling; a smartphone for wirelessly passing speed sensor data and steering direction to computer; an Oculus Rift headset (development kit version) for head tracking and VE display; a Kinect for hand gesture recognition; and a computer for rendering the VE and connecting all the devices with Unity engine. The initial system setup is shown in Figure 3-2. The system and exergames allow player to explore the Street View based virtual world through safe cycling with tricycle, free head movement and immersive virtual environment with VR headset and hand interaction with virtual scene via Kinect.

These applications integrated different hardware devices for studying exercise with VR, which have proved that the immersive experience with virtual reality can increase users’
engagement with physical activity. In this context, our previous VR exergames raised the requirement of a tool for easing the development of future exergaming applications with real-world scenarios.

Although the code of some repetitive functionalities can be reused via copy and paste during the development process of these VR exergames, it still requires a large amount of time investment in developing new applications with the same platform. Thus, based on the literature and the needs that HCI researchers have for developing various VR applications with minimum effort cost but well-designed architecture, it is beneficial and valuable to create an OOAF to lower the complexity in building different immersive VR exergames with reusable functional components and design patterns. The framework will not only help our group but

Figure 3-2. System setup of our first VR exergame (Ijaz et al., 2016b)
also aid other researchers in reusing the design and source code and improving software quality.

Nonetheless, the development of a framework is different from developing an application. A framework is not application itself. Instead, it is an application generator with more complex constructs. In object-oriented software development, there are three major phases (Markiewicz & Lucena, 2001): domain analysis phase (introduced in subsection 3.2) identifies the reusable components of an application domain; framework design phase (described in subsection 3.3, 3.4 and 3.5) outlines the modelled abstraction and novel designs; framework instantiation phase (detailed in Chapter 4) implements the framework software system. The following subsections describe the key aspects in the first and second phases.

**3.2 Domain analysis**

Defined by Prieto-Díaz (1990), domain analysis is “a process by which information used in developing software systems is defined, captured, and organized with the purpose of making it reusable when creating new systems”. As the first phase of framework development, the process of domain analysis includes three basic steps: identification of reusable entities; abstraction or generalization; classification and cataloguing for further reuse (McCain, 1985). Domain analysis helps to capture the domain’s requirements and future requirements. The sources of domain knowledge come from technical literature and existing implementations, our previously built applications, related experiences and documentations, which are used to reveal the requirements and identify the correlated objects.

Our research program aims to explore what motivational drivers are more likely to engage
different type of users in exergames. New treadmills and exercise bicycles try to be more engaging with screens that display a trail or cycling route (NordicTrack, 2017; Peloton, 2017). Although nice scenery is used (e.g. the Tour de France), there are limitations such as these interfaces are not interactive, users cannot look around, stop or do sightseeing. These limitations would reduce the likely users. We sought to explore exergames that had similar scenery but could also have interactive narratives such as tour experiences, or action games. The challenge was that creating the content for such games is expensive and often require large teams of expert game developers.

Summarized from the literature, in exergaming domain, the VR applications should cover the aspects such as: (i) supporting the latest immersive VR technology which are most widely used in HCI community, i.e. HTC Vive and Oculus Rift; (ii) using different safe exercises as full body interaction and physical activity input, especially lower limb movements that are suitable for both young and elderly; (iii) employing interactive virtual environments with user interfaces and minimizing the motion sickness with VR headset to improve the perceived immersion and motivate physical activity; (iv) providing feedbacks for users that not only motivates physical activity but also allows users to adjust movements during exercise; (v) providing engaging game design and content to promote physical activity, etc. Most of the above functionalities are repetitive in different VR exergames therefore can be identified as reusable software components in building framework.

Framework’s requirements are abstractions of similar systems, domain’s requirements and
related experiences. Unlike a software library, a framework not only separates the functionalities but also provides the interaction process between functionalities, for example, the overall flow of control (Dirk Riehle, 2000). In this way, the requirements comprise of encapsulating similar features into one functionality, using different functionalities to represent different software components (input, output, database, etc.), managing the connections between reusable components, providing an extensible and maintainable structure of an application.

Over a number of iterations our team found a theatre metaphor useful in immersive VR exergames. There are several aspects that define a proper VR exergame which can promote players to do moderate physical exercise. The scenery of the game would be like the stage in the theatre. On the stage, actors tell a story. Users are more than spectators, as they can interact with the narratives. Each of this design elements maps to software components introduced in our previous system (Wang et al., 2017) and described in more detail in this thesis. Besides bringing design ideas into a software platform, framework must also be designed to reuse code and software architecture.

As a basic system model to develop specific applications in a certain domain, a framework consists of hot spots and frozen spots (Pree, 1994). The hot spots refer to the flexible parts of a framework which are abstract classes or methods. Technically, hot spots need to be instantiated and implemented with domain-specific code (Mattsson et al., 1999). The frozen spots represent the reusable built-in interfaces or classes that will not change during application development.
**Frozen spots** are the core of a framework which call the *hot spots* written by developer. In other words, *frozen spots* outline the whole architecture of a framework including reusable components and their relationships (Pree, 1995). Object-oriented frameworks can be categorized into white box frameworks and black box frameworks according to the customisation with *hot spots* and *frozen spots* through inheritance or composition.

In the thesis’s case, the software functionalities and VR exergaming practices can be encapsulated into software components with object-oriented programming to build a component-based framework. A component-based framework is often a black box framework for customisation, because the composition approach only requires a user to have a general understanding of the framework and system built rather than a full knowledge of framework architecture required by the white box with inheritance approach (Szyperski, 1998). Thus, the *hot spots* in this framework are relevant to the independent and interchangeable component, where one component encapsulates the code of realizing one functionality of a VR application. For example, the component to support *immersion* and the component to support *embodiment* features. Besides, not only the support for hardware, the *hot spots* can also include encapsulated program module. On the other side, *frozen spots* in this framework are the structure and interfaces that define the relationships between *hot spots*. The components are reusable in developing applications that enhance the flexibility of a framework, and the flexible architecture also allows kernel reuse with *frozen spots*. In addition, the abstractions of both *hot spots* and *frozen spots* should be reliable, extensible and maintainable for future reuse such as
supporting more hardware devices for better exercise experience and feedback, providing more choices of virtual environments for better immersion and interaction.

Following a structure of basic VR system (Figure 2-1), the common architecture of VR exergaming systems which the framework should generate can be summarized in Figure 3-3. Generally, the repetitive functionalities built in different exergames can be encapsulated into framework component with UML and patterns (Booch, 1998; Larman, 2004). Therefore, the input devices (for both immersion and embodiment) including VR devices (head and hand tracking), lower limb exercise devices and exercise feedback devices (physiological device) can be abstracted into independent framework components, as shown in Figure 3-4.

In the same way, the domain specific practices such as virtual scene and database can also be abstracted into interchangeable framework components. The GUI and game mechanics differ from different applications, which are customised through problem specific code written by developers. During the process of generalization in building framework, the abstract classes should have high cohesion and low coupling, which is achieved through using software
3.3 Framework architecture design

Frameworks that support the development of interactive VR systems often provide an integration of different components or modules. Each component or module handles the diverse functionalities of the application such as input, tracking and networking and these can be combined in customised ways for different contexts (Katz et al., 2015; Ohlenburg et al., 2004; Takala, 2014). These frameworks usually follow the composition over inheritance principle in object-oriented programming, in which classes should be polymorphic and reusable by containing instances of other classes for desired functionality, to allow developers to extend the functionality. This principle brings the advantage of greater flexibility in an entity-component-system architectural pattern for game development, where every object in a game’s scene is an entity (e.g. vehicle, bullet, enemy). Each object consists of one or more component
scripts that add the object’s features such as appearance and behaviour (introduced in subsection 2.1.4). This architectural pattern is used in the Unity game engine, which is chosen for developing this framework.

### 3.3.1 Unity game engine

Featuring a component-based architecture design pattern, Unity has become a popular tool for developing real-time interactive systems in HCI. This thesis chose Unity as the platform for VR-Rides framework because it is commonly used among researchers and has a large and active developer community. Additionally, Unity provides features through its core libraries such as rendering, math, networking, scene graph and platform independence which support the creation of both 3D and 2D games. It is also convenient for VR devices (HTC Vive and Oculus Rift) to interact with Unity through manufacturer SDKs which work as Unity plug-ins.

In Unity, the entity is a GameObject, which is used as a container for scripts. The GameObject’s features (appearance and behaviour) are provided by all the component scripts added to it. With this design pattern, component scripts that contain the common functionalities can be shared and reused across different objects. Figure 3-5 illustrates the common structure of a Unity project. The developer adds different GameObjects to a game scene through the Unity Editor, which contain different developer scripts for various appearances and behaviours. The scene hierarchy communicates with Unity Game Engine through Unity core libraries and third-party plug-ins. Basically, the developer needs to work on both GameObjects and scripts to build a game scene, which is repetitive when building similar types of games.
VR exergame development in Unity is challenging because it requires several hardware integrations and functionalities which in turn complicate game mechanic design. Although Unity provides a user interface that simplifies the process of accessing GameObject and component scripts, manual customisation is still necessary to manage GameObject and call functions during runtime. For these reasons, object-oriented method should be adopted to encapsulate common features or behaviours between functionalities so that they can be reusable and extensible.

3.3.2 Proposed architecture

Based on the previously explained motivations, analysis and experiences of building a series of exergames, this component-based framework is designed to simplify the process of developing VR exergames. It assists developers to reuse and extend potentially diverse future VR exergames. In the design phase of this framework, general functionalities and domain specific practices are abstracted and encapsulated into independent and interchangeable components. In the implementation phase, a modular programming pattern and prototype
design pattern are adopted to realize the components as framework modules. Each module contains all the necessary code to execute one functionality. Also, a framework management is used to connect different module components outside of the hierarchy. The reference implementation is written in C# programming language. Figure 3-6 shows the architecture design of this framework.

This framework is layered on top of the Unity scene hierarchy. The framework code interacts with Unity engine through Unity core libraries and third-party hardware via plug-ins. From the perspective of software architecture, there are two layers in this framework. This thesis proposes a manager-behaviour structure in both layers, where behaviour accomplishes basic function and the manager manages the behaviours.

The first layer is the basic component scripts of each module including module manager script and module behaviour script. Both components are derived from Unity’s basic class MonoBehaviour that share and extend the same core behaviour functions. Every module has different numbers of behaviour scripts, where each module behaviour script completes a
single basic behaviour functionality of a module such as converting input data or saving data into local file(s). There is only one manager script in each module, which is used to manage all the behaviour scripts in that module and the data that bridges those behaviours. For example, a manager script defines the common initialization and interchangeability between module behaviour scripts.

The second layer is the core components of framework – modules - that represent different major functionalities. Each module consists of one module manager scripts, various functionality related module behaviour scripts, other function related component scripts and associated GameObjects. In other words, it is a collection of GameObjects – with component scripts attached - that accomplish a module’s functionality. A developer can directly add a module into a Unity scene for specific functionalities through Unity Editor. In this layer, a framework manager script is used to connect, manage and bridge different modules. This component acts as the managing entity and provides a simple way to manage the registered module in scene rather than having the modules managed manually by developer. It is also extensible when more modules are designed and added.

To summarize, in each module, there is one module manager script and several module behaviour scripts, while in this framework, there is one framework manager component and several modules. One module represents the behaviour module for one specific functionality.

3.4 Framework module design

Abstracted from the VR exergaming practices, the VR equipment is implemented in the VR
Device Support module, the sensors and embodiment are in the Physiological & Pedalling Device Support Module. The VR Scene Module implements the staging. There are six modules provided by this framework for a developer to utilize different functionalities in a Unity VR project. Each one stands for one functionality for a VR exergame. Modules are developed into Unity prefabs based on a prototype design pattern - which are predefined templates consisting of various GameObjects and scripts - to create new GameObject hierarchy in Unity scene. Adding a functionality/module into a project can be simply completed through a drag-and-drop operation in Unity Editor (drag the prefab from Project Window into Hierarchy Window and then drop). Each of the modules can be used separately for one functionality, or some modules can collaborate on several other modules. For instance, VR Scene Module can be used together with Module Manager to present a virtual environment in a non-VR project, or in a VR project together with VR Device Support Module. The modules that a developer can choose and combine in implementation are the following.

3.4.1 VR device support module

This module supports the use of modern immersive VR platforms (Figure 2-4) to be integrated into Unity projects. It encapsulates the immersion aspect of VR exergames. The tracking of headset movement and hand controller interaction are provided by the manufacturer SDKs which are used in this module to support both HTC Vive and Oculus Rift. Also, this module contains our scripts to customise button press response and controller user interface, as well as several examples of button functionalities.
The hierarchy in a Unity scene of *VR Device Support Module* is shown in Figure 3-7. It follows the proposed manager-behaviour structure. In this module, the *VRDeviceModule* GameObject is the container of module manager script *VRDeviceManager*, which means that the manager script is attached to this GameObject and can be accessed in Unity’s Inspector Window. It is also the container of three child GameObject components, named *SteamVR*, *OculusVR* and *ControllerFunction* respectively. Each of them consists of different child components as basic module behaviours. Every basic module behaviour accomplishes one part of a parent component’s functionality and the three main components are managed by the *VRDeviceManager* script.

To be specific, *SteamVR* and *OculusVR* are Unity plug-ins provided by manufacturers to support smoothly interfacing these two VR devices with Unity. Both use base stations
(hardware for tracking) to set a room-scale play area, in which the real-time movement of headset and hand controllers are tracked accurately by the stations. In the hierarchy of SteamVR component, the [CameraRig] GameObject defines the tracking area and manages tracked controllers. Its child components Controller(left) and Controller(right) render two controllers in the Unity scene, and Camera(head) replaces the default camera in Unity, which represents player’s eyes in a game. The OculusVR hierarchy contains similar components to enable using Oculus Rift in Unity.

However, the button press responses are left to developers in programming, because these functions are not provided by SDKs. Under this circumstance, this module provides ControllerFunction in its child component to simplify the customisation of controller interactions for both VR platforms. The ControllerEvents component defines the relationship between a physical VR controller and user code, which means that a developer can customise different events to be emitted based on the controller inputs. The other three child components - ControllerToolTips, UI_Interactions and UI_Keyboard - provide examples of controller user interface such as controller button tag, locomotion with buttons and a virtual keyboard which can be directly used, edited or extended in a Unity scene.

Generally, the VR Device Support Module provides a collection of helpful scripts and examples to assist developer adding VR devices into Unity projects. This module covers several common solutions including button interaction, locomotion, interaction with Unity UI and 3D controls. It is a combination of manufacturer SDKs and the abstraction of our domain
practices of immersion in VR exergames.

3.4.2 VR scene module

According to the literature, none of the existing VR exergaming frameworks provided an explorable virtual scene of real-world experiences. Not just exergames, most VR games require the implementation of different interactive virtual scenes by developers, which is a time-consuming work. Thus, this novel design provides the visual content as staging for games. Specifically, it offers the developer an alternative of adding a virtual scene in Unity project without building 3D scenes. This module supports Street View images to be automatically fetched from Google server, and panoramically rendered as the immersive scene to VR headset by Unity (Figure 3-8). Comprising stereoscopic views of countless geographic locations, the staging changes according to customised user interactions (cycling input, button press, etc.). The use of Google Street View allows developers to personalize the environment for different exergaming narratives and individual game player, which is easily done through editing the
initial location(s) in the XML configuration file. In addition, this module also works independently in non-VR projects.

The hierarchy design of this module is depicted in Figure 3-9. Following the manager-behaviour structure, this module has two main components: Head-Up Display (HUD) for fetching and rendering Google Street View images as the panoramic virtual scene, MapController for displaying a mini-map with real-world location in Unity. Bridged by VRSceneModule, these two components use the same XML configuration file for the initial location.

In particular, the HUD component’s basic behaviours are encapsulated into its child components, where Scene is responsible for fetching the Street View images of geographical locations from Google server according to the configuration file and movement signal; LinkMarker calculates and paints the link markers between panorama images, which is the
indicator of street’s direction in Google Street View; LoadIndicator indicates the game player that the next scene of Street View is loading according to movement input; PedalIndicator contains the code to convert interaction input, which comes from customised button press or cycling data, into indicator for scene change. All the behaviours are independent from each other while managed by HUD GameObject, which updates and renders the scene to VR headset. Similarly, in MapController’s child component, Map presents a mini Google map in Unity scene which presents and updates the geographic locations, while Pointer indicates player’s heading direction on the map. Both are bridged by MapController that renders them in scene.

This module provides the functionality of adding an immersive scene with real-world experience to the Unity project, where developers do not need to create their own 3D scenes. The scene can be customised with any places that have Google street views, including many indoor scenes.

3.4.3 Physiological & pedalling device support module

This module abstracts the practical experiences of our previous work and other successful projects to add embodiment features and sensors in exergame development with Unity. our code adds commercial pedal exercisers that output speedometer signal (or common bikes with a commercial speed sensor) as movement input in Unity, which is suitable for both young and elderly to do physical activity in a safe manner. It also supports the data from commercial wristbands (e.g. Fitbit and Microsoft Band) for actigraphy data to be wirelessly passed to Unity for further calculation, display and storage.
Since the VR hand controllers offer a free movement for upper limbs, this module targets to provide exercise for lower limbs in a safe way. Our previous successful experiences (Ijaz et al., 2016a, 2016b) and the exergames by other academics (Bolton et al., 2014; Kikuchi, 2014; Shepherd et al., 2018) reveal that cycling with a stationary device not only avoids the possible stumble caused by HMD cables and the possibility of players getting out of tracking area during immersive gameplay, but also suits both young and elderly. Thus, the exercise hardware supported by this module are two user friendly cycling device shown in Figure 3-10, recumbent trike with suspended wheel, and pedal exerciser DeskCycle (3D Innovations, 2014). The commercial trike does not output any signals, so a wireless bike speed sensor (Garmin Ltd, 2014) is attached to the rear wheel for calculating accurate speed and distance.

With respect to the exercise feedback for players to adjust movements, commercial wristbands provide physiological data such as heart rate and calories for user’s casual exercise status are used in this module. Therefore, this module support two commercial wristbands (Figure 3-11) as real-time physiological data input, i.e. Microsoft Band 2 (Microsoft, 2015a)
and Fitbit Charge 2 (Fitbit Inc, 2016).

The hierarchy of this module is illustrated in Figure 3-12. The `P&PDeviceModule` GameObject manages two child component GameObjects - `CyclingInput` and `PhysiologyInput`. `CyclingInput` processes the data from `DeskCycle` and `Trike`, which converts the raw pedalling data (received from `Middleware Module`) for `VR Scene Module` to use as scene updating indicator. `Physiology` deals with two different wristbands, which extracts and categorizes the physiological data (heart rate, calories, etc.) from XML file.

![Figure 3-12. Physiological & Pedalling Device Support Module’s GameObject hierarchy in Unity](image)
3.4.4 Middleware module

This module includes two parts for supporting the functionality of the Physiological & Pedalling Device Support Module.

On the software side, it provides code to convert and pass speedometer data for use in Unity. Additionally, a mobile application, EMAware (Positive Computing Lab, 2017), is provided for wirelessly passing speed sensor data and wristband data to Unity as JSON (JavaScript Object Notation) file.

The hardware involved in this module are shown in Figure 3-13. The Arduino Uno (Arduino, 2012) is a microcontroller board to pass the speedometer data to Unity. It connects speedometer with a 3.5mm audio cable to receive data and PC with a USB cable to pass the converted data. The Android phone runs our EMAware application to fetch real-time sensor data from wristbands and wirelessly pass the data to our server, which can be reached in Unity for downloading as JSON file.

Figure 3-13. Arduino Uno microcontroller board (Left) and Android phone (Right)
In Unity, this module’s hierarchy has two child component GameObjects (Figure 3-14), *PhysiologicalData* and *PedalingData*. Each of *MSBandData* and *FitbitData* GameObjects has the code to fetch the wristband data from our server (in JSON file) and converts it into XML file, while both *SpeedometerData* and *SensorData* can read the cycling data from USB port and application, and convert the data.

### 3.4.5 User study module

This module generalizes the features of designing user studies in Unity project for HCI research. For example, with this module, developers can set dependent variables and parameters to be automatically recorded in this module, as well as grouping different experimental design types (e.g. *Between-Subjects*: one subject is exposed to only one condition and contributes one entry to the whole dataset. Subjects are randomly allocated to one of the conditions. *Within-Subjects*: each subject is exposed to each of the conditions, contributing one entry for each of the conditions).
The hierarchy design of this module is shown in Figure 3-15. There are three child components managed by UserStudyModule, which are PlayerInfo, PhysiologicalData and GameplayParameter respectively: PlayerInfo contains script to log in player’s information; PhysiologicalData contains the script to fetch, convert and categorize the physiological data from Middleware Module; GameplayParameter contains script to set exercise goal and display gameplay related status.

More specifically, the Inspector Window in Unity Editor of three scripts - PlayerInfo, GameplayParameter and UserStudyManager - are designed and shown in the following Figure.

According to Figure 3-16, the basic behaviour of PlayerInfo script is to log in player’s information before gameplay. The input of PlayerID, Gender, Weight, Height and Age can be
directly done in Unity’s Inspector Window, which also automatically calculates the body index such as BMI (Body Mass Index) and BMR (Basal Metabolic Rate). The GameParameter script has two basic functionalities: set exercise goal, i.e. target time to cycle, target distance to cycle and target calories to burn; and calculates gameplay status such as time elapsed, distance cycled, real-time speed and geodistance (according to Google Maps in VR Scene Module). UserStudyManager bridges all the base behaviours, in which the grouping method for user study can be chosen, as well as the parameters from child components to be saved (ticked parameters to be saved during gameplay automatically).

### 3.4.6 Database module

![Database Module Design in Unity](image)

Figure 3-17. Database Module Design in Unity

To handle the data exchange between different modules at runtime, a general database is
required. This framework uses XML as the data format, since Unity uses C# in programming that supports XML schemas well.

As shown in Figure 3-17, the design of Database Module includes the basic data model structure, I/O data structure and database. In detail, the reasons of this design are following:

(i) the basic data model includes the basic information provided by all the modules in this framework, which includes physiological data and pedalling data from Physiological & Pedalling Device Support Module, player information and gameplay data from User Study Module. All the data from one user/player have the same GameID as primary key, representing the object that all the data belong to. Besides the basic data, other customised data by developer using this module can also be added using the same structure, with their foreign key referring to primary key.

(ii) the data format used in every module is XML document. XML document has different tags that offer extra information on sections of the document, in which a user can add new tags meanwhile separately specify how each tag should be handled. This feature increases the extensibility of XML. The ability to specify new tags and create tag structures makes XML a general method in data exchange and structured data sets. In addition, XML is also simple and human readable. Therefore, XML has the advantages in extensibility, generality and simplicity. On the other hand, it is one of the standardized formats for data that is commonly used in game development.

(iii) According to the above facts, the thesis designs an XML-based database and data
pipeline with C# containers for data exchange, storage and management, which is independent from Unity features. With the benefit of XML in supporting complex data structures, the XML-based database provides the functionality in separating data from code, which is easier for developer to edit data files. Moreover, developer can save game related data in a compact form and add new entities into game with the use of XML file and XML-based database. Additionally, XML is platform independent, so that this module also supports developing applications that uses both client and server written in C#.

In summary, this module uses XML document as data format and an XML-based database for storage and data exchange. It provides the code for XML serialization and deserialization to benefit users with the advantages from XML. The data structure further simplifies the exchange and storage of customised user study oriented data.

3.5 Framework data flow

Based on the Entity-Component System architectural pattern, this framework code aims to make the source code as modular as possible through a clear separation between data and behaviour. In this case, behaviours are included within each module and interactions between modules are through data. The framework uses XML serialization and deserialization for processing data, which means that the data exchange between modules are through XML format.
The data flow and relationships between modules in a Unity VR project using this framework are depicted in Figure 3-18. Each exergame has entity data that comes from game devices. In this case, the data is interaction data from the VR headset and controllers, pedalling data from pedal exerciser and bike, and physiological data from wristband. The data from VR devices is handled by the interaction behaviour and manager script, which uses manufacturer SDKs to allow headset and controllers to interact with Unity scene. Meanwhile, the exercise data is dealt with by the Middleware Module: cycling data from the pedal exerciser is converted into speed (indicated as revolution per second) and passed to Unity via the Arduino microcontroller script. In addition, if a bike with a speed sensor is used for movement input, the speed sensor data is converted and wirelessly passed to Unity by our Android application. Physiological data such as heart rate and calories are converted into JSON files and wirelessly passed to Unity through our Android application. In Unity, exercise data and pedalling data are processed in the Physiological & Pedalling Device Support Module, which allows the data to be used for both scene interaction and database storage. User study parameters are also passed.

Figure 3-18. Data flow of the framework in terms of modules
to Database Module from the User Study Module. All the modules can be managed through the Framework Manager. Consequently, when the framework is used, the developer needs only to write the game mechanic classes that use the interaction data and exercise data to build the exergames.
Chapter 4. Framework Implementation

Summary

After presenting the design specification of framework architecture and framework modules, the next step is to implement modules as software components (Unity prefabs) that are easy for developers to use. This chapter introduces the implementation of module’s basic class extending Unity’s core event functions. Then the detail implementations with design patterns used on each module’s structure including behaviour classes and manager class are presented. This chapter extends and takes partial description from the two published papers (Wang et al., 2017, 2020) listed in the Preface section, where I am the lead author.
4.1 Introduction

Since the framework is based on Unity game engine, the implementation stage is done through using Unity Editor for GameObject hierarchy editing and Microsoft Visual Studio 2015 for C# scripting.

In Unity Editor, every object in a game scene - such as character, tree, light and audio source - is a GameObject, which can be added and managed in Unity’s Hierarchy Window (Figure 2-10). A GameObject’s properties are determined by the components attached to it. In other words, a GameObject can be seen as a container for component scripts. Meanwhile different component scripts add different features to this GameObject. The component scripts can be added and managed in Unity’s Inspector Window (Figure 2-11), which also shows the public variables of the scripts.

The scripting in C# with Unity involves the base class of Unity scripting API, MonoBehaviour, which all the component scripts of a GameObject that execute during runtime should derive from. In MonoBehaviour class, there are several commonly used core event functions, Awake(), Reset(), Start(), Update() and Destroy(). For instance, Awake() runs when the component script instance is being loaded, so it act as constructor and is used for the initialization of variables and game state; Reset() is called after Awake() when the script is attached and not in play mode; Start() is called only once after Awake() and also used for initialization on the frame when the component script is enabled. Update() runs every frame where MonoBehaviour is enabled, therefore it is the
function where the developer specifies the behaviour or game logic code. Destroy() removes the GameObject or component. Besides, though there are many other core event functions such as OnEnable() for initialization, FixedUpdate() for frame-rate independent message on physics calculations, OnGUI() for rendering and handling GUI events, and OnDisable() for decommissioning, the above five functions are mainly used in the framework module scripting. A sample of new class that is generated in Unity Editor and derived from MonoBehaviour is shown in Figure 4-1.

When a script derived from MonoBehaviour is attached to a GameObject in the Unity scene, the event functions in this script will execute based on Unity’s script lifecycle flowchart (Unity Technologies, 2017) after the scene starts. The simplified execution order of MonoBehaviour’s main event functions which are commonly used in Unity is shown in Figure 4-2. In this flowchart, the physics cycle and GUI rendering cycle may execute multiple times per frame according to developer’s parameter settings.
4.2 Basic class implementation

In Unity, one GameObject can contain different numbers of component scripts derived from MonoBehaviour to execute during runtime, in which each component script adds different basic behaviours or features to this GameObject. In Unity’s scene hierarchy, a GameObject can have different GameObjects as child components, in which every child GameObject stands for one major function of this parent GameObject. In this case, the child GameObjects’ transform (position, rotation and scale) follows the change of parent GameObject’s transform.

With the Entity-Component-System design pattern, this framework’s modules are a collection of component scripts (Figure 4-3) derived from MonoBehaviour with associated
Figure 4-3. UML class diagram on the relationship among Unity's base classes (MonoBehaviour and GameObject) and our framework module classes (module behaviour and manager)

GameObject. In Figure 4-3, GameObject is the fundamental entity in Unity scene and GameObject is the base class for all entities in Unity scripting API. GameObject class has the properties such as activeInHierarchy for defining whether GameObject is active in scene, scene for indicating the scene that the GameObject is part of, transform for indicating the transform (position, rotation and scale) attached to this GameObject, etc. It also has public methods for developers to call, for example, GetComponent() returns the
component if the GameObject has one attached, `SendMessage()` for calling the method on every `MonoBehaviour` in this GameObject and `SetActive()` for activating/deactivating the GameObject in a game scene.

With respect to the framework, there are two kinds of component scripts in each module: module behaviour (contained in child GameObject) and module manager (contained in parent GameObject). Module behaviour receives data, converts data for modules to use, or accomplishes one basic functionality, where one module could have different behaviours. Module manager deals with data from behaviours to complete the functionality of this module, where one module contains only one module manager. All the functionalities are encapsulated in behaviour classes in programming through abstract factory design pattern, which brings reusability to framework’s code.

4.3 Module implementation

To encapsulate the functionalities of each design aspect, the implementation of modules used the manager-behaviour structure mentioned in the framework design of Chapter 3, where each module has one manager script to bridge and manages multiple module behaviour scripts. Since every module’s scripts are contained in parent and child GameObjects of this module, the class hierarchy of each module is the same as the GameObject hierarchy introduced in the design (section 3.4) of Chapter 3. This chapter introduces and specifies the implementation stage of all the modules’ main classes and the framework.
4.3.1 VR device support module

According to the GameObject hierarchy of Figure 3-7, the **VRDeviceModule** contains **VRDeviceManager** script and three child component GameObjects - **SteamVR**, **OculusVR** and **ControllerFunction** - with lower level GameObjects in each containing basic behaviour scripts. The UML class diagram of this module is illustrated in Figure 4-4.

As shown in Figure 4-4, there are three major parts in this module where each part stands for a subfunction of this module.

The first part handles HTC Vive, in which the manufacturer SDK SteamVR (Valve Corporation, 2015) is included in the child components. There are four main scripts in this SDK that are derived from **MonoBehaviour** and need to be attached to GameObject in Unity Scene for execution during runtime: **SteamVR_PlayArea** for drawing different sized room-scale play areas; **SteamVR_ControllerManager** for enabling/disabling objects on connectivity and assigned roles; **SteamVR_TrackedObject** script for controlling in-game objects with tracked hand controllers; and **SteamVR_Camera** for adding SteamVR render support to existing camera objects in Unity. In this module, these four scripts are contained in child component GameObject **[CameraRig]**, **Controller** and **Camera** separately (shown in Figure 3-7), where the class hierarchy is same as the GameObject hierarchy.

The second part uses OVR SDK (Facebook Technologies, 2017) to support Oculus Rift. Similar to SteamVR, the main classes derived from **MonoBehaviour** are separately associated with child components, in which the class hierarchy maps to the GameObject
CHAPTER 4. FRAMEWORK IMPLEMENTATION

The three main classes (demonstrated in Figure 4-4) that contained in child component for providing the core behaviour functions are: OvrManager for configuring Oculus VR device related data and parameters; OvrCameraRig for adding head-tracked stereoscopic VR camera rig in Unity Scene; and OvrAvatar for adding headset and controller...
render support to camera in Unity.

The third part is my original code for simplifying the customisation of controller button press response for both VR platforms, as well as some extendable sample functions. All the classes inherit and extend the core event functions in MonoBehaviour. For instance, Events extends the Update() function through adding extendable functions to provide the response for different controller button press, in which listeners of different controller events are specified. Virtual method in programming is used in this class, so that it is easy for developers to override the events to be emitted based on the controller inputs; Tooltips allows the developer to customise tags for controller buttons that is attached to controller and moves together with controller, which is shown in Figure 3-8. This function helps developers to easily add hints on VR controllers for game players; UI_Interactions provides an example of virtual canvas UI that can be interacted with controller button press, where the controls on the canvas can be customised by developers; similarly, UI_Keyboard provides a virtual keyboard UI example for developers to easily add into Unity projects, in which a player can use VR controllers to type the virtual keyboard for letter input. In total, this part not only supports a customisation of controller button events, but also provides customisable templates for virtual UIs, which are a collection of practices from our previous VR projects and studies.

These three parts including the basic behaviours are bridged by VRDeviceManager, which provides developers the option for specifying the VR platform used (either HTC Vive or Oculus Rift) in Unity’s Inspector Window. In addition, the ControllerFunction part
supports both VR platforms, especially that the Events script includes the functions supporting both HTC and Oculus VR controllers on button initialization and customisable responses for different button press events.

The classes in Figure 4-4 are the main classes of this module that all derive and extend the core event functions from MonoBehaviour and need to be added into Unity Scene for execution during runtime. Besides, other scripts which are not derived from MonoBehaviour are not included in the figure but need to be in the project’s folder to work as dependency.

4.3.2 VR scene module

Figure 4-5 shows an example of the VR Scene Module implementation. According to the figure, main classes (inside the dashed line box) are contained in GameObjects as Unity prefab (a template for creating new prefab instances in Unity Scene).

In order to be able to change the virtual scene and render the scene to Unity camera based on interaction input, this module is comprised of four behaviours for four functionalities. All the behaviour scripts are contained in the GameObjects that have the same names listed in the following: (i) Scene class realizes the functionality of fetching the panorama images of given location from Google server; (ii) LinkMarker calculates the geolocation, paints and updates the link marker between panorama images in Unity scene, which is the indicator of a street’s direction in Google Street View; (iii) LoadIndicator tells the game player that the next scene of Street View is loading based on interaction input; (iv) PedalIndicator contains
the code to convert interaction input – which comes from button presses or pedalling data – into indicator for scene change. All the behaviours are independent from each other while managed by `SceneManager` in `VRSceneModule`.

In detail, every panorama image fetched from Google server consists of six sub-images labelled as front, back, up, down, left and right. These six images represent the six sides of a cube wrapper. In Unity, this cube is called `Skybox`, which is a wrapper around Unity’s `Scene` that shows what the world looks like beyond the geometry. In other words, all the GameObjects and Unity Scene are inside this `Skybox`. The six sub-images are wrapped to the six sides (also called front, back, up, down, left and right in Unity) of this `Skybox` to present a panoramic view...
of Google Street View with Unity Camera.

Besides, the instance of Panorama in Scene class receives the information of a panorama image from Google server, which contains the location’s latitude, longitude, six sub-images, panorama link between panorama images and so on. The link marker in Google Street View links two adjacent panorama images. In this case, it connects the front sub-image of current location with the back sub-image of next location. This module’s LinkMarker class uses this information and PanoramaLink class to paint the link markers to inform players the direction of streets (the direction he could head to).

In addition, the panorama image can refresh by loading the images of next location and link marker, according to the signal of loading next panorama image. In this module, the signal is processed by LoadIndicator. In SceneManager, the instance of LoadIndicator generates signal of loading next scene according to the interaction data passed from the instance of PedalIndicator. The signal is used by the instance of Scene and LinkMarker, which fetches new panorama images and new link marker based on Google Street View. SceneManager renders the ‘Loading next scene’ text and the new scene to Unity’s Skybox, which is a 360-degree panoramic view in VR headset.

The behaviour classes and manager class are all derived from MonoBehaviour and contain the implementation of same event functions such as Start() and Update(), so they need to be in the Unity Scene (through attached to GameObjects in scene) for execution in every frame of a Unity game.
To summarize, the main classes contain all the necessary code to accomplish the functionality of providing a dynamic panoramic virtual scene based on Google Street View images that change according to player’s input. A programmer only needs to import the VR Scene Module’s prefab and add it into the scene hierarchy through Unity Editor, then configure the VR camera to enable this module in a Unity project without programming. Furthermore, not only can the Unity prefab (which is a set of GameObjects) be reused in Unity projects, every single behaviour class in this module can be reused in future projects.

4.3.3 Physiological & pedalling device support module

The implementation of Physiological & Pedalling Device Support Module is based on the hierarchy introduced in subchapter 3.4.3. Contained in the associated GameObjejcts in Figure 3-12, this module has two parts for two functionalities and one module manager script for bridging them. The main classes involved in the module’s structure are shown in Figure 4-6.

The first part deals with the cycling hardware. As illustrated in Figure 4-6, the CyclingInput script and its directly associated behaviour scripts, Datalog and PedalProgress, are included as components in CyclingInput GameObject for converting the cycling data from Middleware Module (raw impulse signal from sensors) into the data format for other modules to use.

In particular, the CyclingInput script is responsible for receiving the data from the manager script in Middleware Module through the instance of Pedal() class and using PedalProgress class to convert the data. The converted data is indicated as revolution per
second and pedal progress, which is then used in *VR Scene Module* as the signal for scene change and player movement. Meanwhile the data log for cycling is handled by this part for displaying cycling progress in game.

The second part handles the physiology data from wristbands, where the scripts are contained as components in *PhysiologyInput GameObject*. According to Figure 4-6, the *PhysiologyInput* class receives the raw wristband data from *Middleware Module*, converts the data from JSON file to *PhysiologyData format* and uses *DataFilter class*
to categorize the data into heart rate, calories, skin temperature, etc.

In summary, the P&PDeviceManager class in P&PDeviceModule manages these two parts. With this manager script, the developer only needs to select the hardware used (DeskCycle, trike, Microsoft wristband or Fitbit wristband) through Unity’s Inspector Window and the basic behaviours such as data receiving, data transferring, data filtering and data logging will automatically execute when the project runs in Unity.

4.3.4 Middleware module

Similarly, the software perspective implementation of Middleware Module is based on the GameObject hierarchy shown in Figure 3-14. This module contains MiddlewareManager script and two child component GameObjects – PhysiologicalData and PedalingData, which comprise the basic behaviour scripts in their lower level hierarchy. Figure 4-7 demonstrates the main UML class diagram of this module.

Since this module is designed to fetch the raw data from hardware and then pass the data to other modules, the class hierarchy consists of two parts. The first one handles the cycling hardware. On one hand when DeskCycle is used, the basic behaviour class Speedometer is implemented for reading the cycling data from Arduino microcontroller via serial port. Another basic behaviour class is SerialData, which connects and initializes the serial port that links Arduino. It also reads the serial messages sent by Arduino as pedalling signals. On the other hand, when the speed sensor is used with trike, the Sensor class in this part provides the functionality of fetching the data from our server, which is wirelessly uploaded by our mobile
The second part deals with the physiological data collected with wristbands. Within this part, the Physiology class uses the DataRequest behaviour to obtain the JSON file - which contains all the physiological data sent from our mobile wristband application (EMAware) - from our server. The data format related classes PhysiologyData and Band application.

![Figure 4-7. Main classes in Middleware Module](image-url)
are used in this method for requesting and converting data.

With the manager-behaviour structure in every module, this module employs MiddlewareManager script in parent hierarchy to manage both parts. It saves the developer’s effort in writing code to pass the data from hardware to Unity by simply choosing the device used with this module in Inspector Window.

4.3.5 User study module

The User Study Module is implemented according to the hierarchy and design aspects introduced in subchapter 3.4.5, with the basic behaviour scripts included in related GameObjects. This module has three components for supporting the user study oriented VR exergame development. Main classes derived from MonoBehaviour in the components are shown in Figure 4-8.

Mapping the hierarchy design to the class implementation in Figure 4-8, there are three main functions in this module: (i) the PlayerInfo class implements the functionalities of inputting player related information (e.g. anonymous PlayerID assigned, gender, weight, height, etc.) through user interface in Inspector Window. It also provides the functionality of calculating the basic index of a user such as BMI and BMR according to these data, which can be further used in further evaluation of VR exergames built. In addition, it further supports the involved data to be stored as XML files. (ii) the GamePlayParameter script provides user interface in Inspector Window for customisation, which includes setting exercise goal, displaying and recording game play status into XML files. (iii) this module also includes the
CHAPTER 4. FRAMEWORK IMPLEMENTATION

class (PhysiologicalData) for fetching the physiological data from Middleware Module, in which the data is filtered into different categories and recorded respectively into XML files.

On top of the hierarchy is the UserStudyManager in UserStudyModule GameObject. This script allows an application user to choose the grouping method of a user study (i.e. between-subjects or within-subjects), which leads to different data storage approaches such as folder names, file names and the data types in each file. It also allows user to choose which types of data (including player profile, player index, physiological data and gameplay data) to be recorded through toggle control (checkbox) in Inspector Window, where all the data are chosen to store by default.

Figure 4-8. Main classes in User Study Module
In conclusion, the User Study Module not only eases the implementation of user study oriented applications through the functions provided by its scripts, but also simplifies the usage of application in experiments where data associated operations are handled by this module through operations in user interfaces.

4.3.6 Database module

The data structure and database design on Database Module has been introduced in subchapter 3.4.6. Based on the requirements of other modules and the usage scenario of this framework, there are few functionalities implemented in this module: 1) loading the parameters, settings and initialization related data from XML file and XML file contained in Unity assets; 2) saving the data as XML file into asset folder during runtime; 3) supporting data storage and data exchange between modules during runtime. In particular, all the modules use XML serialization and deserialization in data processing for data exchange and database accessing, which is extensively used in game engine for storing and transferring data.

Thus, the implementation of this module is shown in Figure 4-9. In the implementation, this module can load configuration data contained in XML document in resource folder during initialization process of application. With the use of Unity, all the data from other modules are defined in C# and serialized in XML during runtime, as well as stored in database.

With respect to the data loading, this thesis uses AssetBundle for resource loading, which is defined by Unity as an archive file to load at runtime. It contains non-code assets, including not only configuration data but also game resources such as models, textures and audio clips.
Therefore, this module adopts AssetsBundle and XML database for data loading, in which the reference information of data are implemented in Figure 4-10. In this implementation, the DatabaseEntryRef data type allows entries in database to reference each other; the AssetBundleInfo class contains the filename and path of each; the AssetInfo class includes the description of a bundle, the reference to this bundle, and the path to this asset.

The loading behaviour script uses the information defined in the above classes and Unity’s core library functions LoadAssetAtPath() for accessing data or asset from editor. These classes build a pipeline for database to asynchronously load resource through XML. A developer only needs to specify the information about assets and bundles when using this approach without any metadata involved. In addition, it is easy for developers to add new data or asset bundles through defining XML entries.

In summary, the Database Module combines the AssetBundle with XML document to build an XML-based database. This module provides a simple and platform independent way for source control with XML. The AssetBundle pipeline separates the data and code, allowing
developers to set up and modify configuration data, textures, prefabs or models without using Unity. In addition, adding new entries and loading new data files are more efficient with this method and database.

All the framework modules are implemented in the same way, where detail functions are realized in different module behaviour scripts and all the behaviour scripts are bridged via the module manager script. If a new functionality needs to be added to extend one module, the developer only needs to: realize the function in a new module behaviour script; then deal with the behaviour interaction in the module manager script; and then attach this behaviour script to the module’s GameObject. In framework’s layer, major functionalities are realized into

```csharp
using UnityEngine;
using System.Collections;
using System.ID;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

public sealed class AssetBundleInfo : DatabaseEntry
{
    [XmlElement]
    public string Name { get; private set; }

    [XmlElement]
    public string URL { get; private set; }
}
```

```csharp
using UnityEngine;
using System.Collections;
using System.ID;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

public sealed class AssetInfo : DatabaseEntry
{
    [XmlElement]
    public string Path { get; private set; }

    [XmlElement]
    public DatabaseEntryRef<AssetBundleInfo> AssetBundleInfoRef { get; private set; }
}
```
different modules and all the modules are managed through the framework manager script. In this way, it is easy to manage and extend each module’s functions and framework’s modules.
Chapter 5. Framework Code Evaluation

Summary

After the implementation of the framework features that are described in previous chapters, the next step is to build applications using this framework and evaluate how this framework improves the development of VR exergames. This chapter evaluates the framework from a software engineering perspective by comparing the VR exergames we previously built for ethics approved user studies through code metrics studies and analysis. Results show that this framework reduces the effort for researchers in VR exergame implementation and improves development quality. The results also suggest that the framework is reliable and maintainable. This chapter extends the description from the two published papers (Wang et al., 2017, 2020) listed in the Preface section, where I am the lead author.
5.1 Introduction

Since the goal of this framework is to reduce development effort, increase development efficiency and quality through the abstraction of domain knowledges and object-oriented designs, we need to verify the effectiveness of this framework in practices. Therefore, several VR exergames were then developed using this framework to conduct ethics approved user studies in exploring the performance of exercise with VR in different designs.

To evaluate the performance of framework in assisting building VR exergames, standard indexes need to be calculated to provide the assessment of software development. According to the experience from government and industry, Schultz (1988) introduced a set of software metrics - e.g. source lines of code (SLOC) as software size metric and cyclomatic complexity as design complexity metric - to monitor and reveal the effort cost in the progress of software development. Based on his report, this thesis evaluated the proposed framework with several software metrics using standard tools. Evaluation studies and analysis are described and discussed in the following subchapters.

The effectiveness of framework in assisting HCI research is evaluated through the development of the VR exergames. In this chapter, the framework is evaluated from two perspectives via six VR exergames that we previously built. These will be used for comparison and analysis (introduced in subchapter 5.2).

The evaluation first compared the code efficiency of the framework between our VR exergames built with and without using the framework through three code studies, based on
CHAPTER 5. FRAMEWORK CODE EVALUATION

the code metrics related to software development progress (demonstrated in subchapter 5.3). Second, the code metrics of the framework that influence software development progress is analysed, which comprise of a set of software measures for code reliability and maintainability (illustrated in subchapter 5.4).

5.2 Applications for evaluation

This subchapter introduces the VR exergame applications that are used in evaluation of the framework. There are six VR exergames in total involved in this chapter, in which students in our lab built each of them for conducting ethics approved user studies. The exergames were built on the same computer and in the same software environment, as well as the code study environment and code analysis environment during evaluation. To be specific, the implementation and evaluation analysis both run on Unity game engine (version 5.6.0f3, 64-bit), Microsoft Visual Studio 2015 and 64-bit Windows 10 Operating System.

- **Game 1 and Gam 2**

In 2016, we developed two different VR exergames from scratch to conduct our first user studies. Two PhD students, without any experience in Unity or virtual reality, built a VR exercise platform (Ijaz et al., 2016b) and two VR exergames (Ijaz et al., 2016a) to understand how different game designs impact user experience in a VR exergaming environment. The VR exercise platform setup is shown in Figure 3-1. On the hardware side, it includes an Oculus Rift headset (development kit version), a stationary trike, a smartphone and a Kinect. On the software side, it uses Unity game engine to render the virtual scene, process the hardware
interaction and game mechanics. During the development, 320 man-hours and 240 man-hours cost were respectively logged in the producing stages for Game 1 and Game 2.

Particularly, Game 1 was a competitive guessing game. In this game, the player could cycle in different unknown locations within different cities. There were in-game audio cues and map to navigate the player to find some nearby landmarks. Successful guess would give scores to player and unlock more difficult levels. There were features such as in-game scoreboard and leader board (based on time and score) to foster the sense of competition. Game 2 was an affiliative tour game. In Game 2, the player could use hand gestures to interact with the virtual UI to choose three famous cities for virtual tour. The sightseeing tour allowed player to cycle in virtual city streets comprised of Google Street Views and listen to the introduction audio of major landmarks. Players could also navigate themselves according to in-game maps and take snapshots via hand gestures with the help of Kinect. In addition, the photo album generated in the tour can be shared with friends and family. This affiliative game did not require any competence or performance goal, compared with the competitive game.

These two VR exergames were developed without using framework. At a later stage, the games were updated, where HTC Vive was used to replace Oculus headset and Kinect because it provided better HMD and hand interaction experiences.

- **Framework**

With the intention of increasing code reuse and design reuse in future development, and providing students without experiences in VR with proper tools to build VR exergames, I built
this framework with a 450 man-hours investment in design and implementation in the time period of 2016 to 2017.

- **Game 3 and Game 4**

Based on the framework and same hardware setup, two VR exergames (Ijaz et al., 2017) were built by two students in 2017 to conduct ethics approved user study on the enjoyment of physical activity on a VR exercise platform. Both students were using Unity and VR for the first time.

In the development of these two games, part of the framework modules were used, where Game 3 used all modules except for *VR Scene Module* and Game 4 used all the modules. Both games used a same hardware setup including HTC Vive, a smartphone, a trike and a wristband. In the design aspect, Game 3 was a relatively simple cycling experience with exercise feedback, in which the player could cycle within a static virtual view with exercise status (real-time heart rate, calories burnt and activity time) displayed on virtual UI. Game 4 was a cycling game with explorable immersive virtual streets, in-game map for navigation, and real-time exercise status (heart rate, calories, time, distance, etc.) displayed as feedback. During the development of these two VR exergames, 70 man-hours and 80 man-hours were invested respectively.

- **Game 5 and Game 6**

Furthermore, in our latest study, another two exergames (Ijaz, Wang, et al., 2019) were developed by one student learning and using this framework in 2018. A new system setup was adopted, and new game mechanics were created. Specifically, the hardware involved HTC Vive,
DeskCycle, Microsoft wristband and a smartphone.

In this study, a VR balloon shooting game (Game 5) was built, in which a player could cycle in a virtual city with DeskCycle, look for hints and navigate himself with in-game map, and find the floating balloons to shoot with longbow using VR controllers. Balloons varied in floating speed, colour and size, which provided different scores if shot. Game play status including score, activity time, distance cycled, physiological data and leader boards were displayed on virtual UI. Another game (Game 6) was a VR Pokémon catching game. In this game, the player could cycle to explore a virtual city, find the special places where Pokémons show up according to hints on map, and catch the Pokémon through throwing Poké balls towards them. Game play status similar to Game 5 were also displayed in this game.

Both Game 5 and Game 6 were built using all the modules in the framework to explore motivation and enjoyment factors in different VR exergaming designs. A total of 120 man-

<table>
<thead>
<tr>
<th>Time period</th>
<th>Project</th>
<th>Experience with VR</th>
<th>Using framework</th>
<th>Working time (hours per day)</th>
<th>Total investment in development (man-hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>Game 1</td>
<td>N</td>
<td>N</td>
<td>6-8</td>
<td>320</td>
</tr>
<tr>
<td></td>
<td>Game 2</td>
<td>N</td>
<td>N</td>
<td>6-8</td>
<td>240</td>
</tr>
<tr>
<td>2016-2017</td>
<td>Framework</td>
<td>Y</td>
<td>/</td>
<td>6-8</td>
<td>450</td>
</tr>
<tr>
<td>2017</td>
<td>Game 3</td>
<td>N</td>
<td>Y</td>
<td>5</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Game 4</td>
<td>N</td>
<td>Y</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>2018</td>
<td>Game 5</td>
<td>N</td>
<td>Y</td>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>Game 6</td>
<td>N</td>
<td>Y</td>
<td>5</td>
<td>100</td>
</tr>
</tbody>
</table>
hours and 100 man-hours were invested by the student for each game respectively (base on student’s milestone report). The student had no prior experience with virtual reality.

Based on the registered development schedule, project timetable and student reports during all the projects, the detailed investment of each student in project development is summarized in Table 5-1.

5.3 Framework code efficiency study

In the framework code efficiency study on the progress metrics that is related to effort cost in development, the thesis compares how this framework helps developers’ code applications for the above six games developed. Even though the Unity Editor can save a lot of work for a developer to build scenes, GameObject hierarchy and components, code writing is still the core activity in game development, especially in game mechanics design and game data processing.

In this evaluation, this framework code efficiency study only included the C# code provided by the framework, developer’s C# code and developer’s configuration code (if applicable) in each Unity project. Other Unity APIs, plug-ins and Unity packages such as hardware device manufacturer SDKs are excluded. Three code studies focusing on source lines of code, Intermedia Language (IL) code and cost in man-hours are discussed in this subchapter.

5.3.1 Code study 1

As SLOC are generally used to estimate the effort required in developing an application because it directly reflects the necessary effort to build a system or application as a software
code size metric (Albrecht & Gaffney, 1993; Schultz, 1988), this study compared the total SLOC written by the developers and those provided by the framework for the six VR exergames. Game 1 and 2 did not use framework while Game 3, 4, 5 and 6 were developed with part or all modules of the framework. This evaluation used a configurable counting tool which not only excluded the blank lines and comment lines but also used a set of counting standards (Nguyen et al., 2007).

Table 5-2 and Figure 5-1 present the count results of total SLOC written by the developers and provided by framework in the development of above six games. According to Table 5-2, Game 1 and Game 2 were built without the framework, in which the developers wrote 2968
and 4167 SLOC for Game 1 and Game 2 respectively. Game 3 used part of the modules and had a simple game mechanic design, so it required only 562 SLOC for the developer to finish, where 1284 SLOC from the framework making up 70% of the total SLOC for Game 3. Next, Table 5-2 shows that for Game 4, 5 and 6, where the game designs required more code because of more gaming features involved, developers needed to write much less code to finish the games. This is because that the framework provided all the reusable code for the module functions. In this case the developers did not have to do complete repeat this code writing to realize those functions.

Although the lines of developer code and framework code differ from each game due to different game mechanic designs, it can be seen from Figure 5-1 that the framework code in Game 3, 4, 5 and 6 takes up a significantly large proportion in programming (namely 70%, 79%, 79% and 76% respectively). Compared with Game 1 and Game 2, where all the code was written from scratch by developers, the significant advantage of employing the framework in the other four games is that it helped to reduce at least 70% of the programming effort. Furthermore, because the games were getting more complicated due to complex game features, there is an increase in developer code in Game 3, 4, 5 and 6, yet the framework code contribution remains above 70%.

5.3.2 Code study 2

Considering only the source lines of code does not fully reflect the advantage of using framework in code writing. In object-oriented programming, the size of executable code is not
a negligible index as it also serves as a metric for measuring the size of the software (Necula et al., 2002). In C#, Intermedia Language (IL) is the language code generated by the C# compiler that is executed during runtime. Hence, this study compared the lines of IL code from developer’s code and the framework’s code for the six exergames in order to analyse whether the framework helped to reduce programming work. Microsoft Visual Studio 2015 was used as the tool to run the standard code analysis for calculating this index.

Table 5-3 and Figure 5-2 show the lines of IL code provided by the developer and this framework in each of the six games. Game 1 and Game 2 were developed without framework,
which had 1239 lines and 1751 lines of IL code respectively. As observed, in the design of Game 3, which had a simpler game mechanic design compared with Game 4, 5 and 6, the IL code by the developer was 140 lines, while the IL code by framework was 561 lines (which consists of upward of 80% of IL code). Meanwhile in Game 4, the lines of IL code by developer and framework were 260 and 1071 respectively, where the contribution of framework IL code made up of 80%. Further, in Game 5 and Game 6, where both had a complex game mechanic design, the proportion of developer’s IL code were 21% and 24% respectively and those of framework IL code were 79% and 76% respectively. Additionally, in the game designs of Game 4, 5 and 6 (that need more programming work due to the game features), the framework code reduced more than one thousand lines of IL code, which means that the framework code reduced a large amount of the software size required in code writing. According to Figure 5-2, the framework code took up the major of code size, namely 80%, 80%, 79% and 76% in Game 3, 4, 5 and 6 respectively. It can be concluded that with the help of this framework, the development effort in these VR exergames was significantly reduced.

5.3.3 Code study 3

As man-hours is a commonly index used in software cost estimation (Boehm, 1987), this study compared the cost in man-hours of developing the above games (listed in Table 5-1). The investment in games developed with framework and those without framework are depicted in Figure 5-3.
CHAPTER 5. FRAMEWORK CODE EVALUATION

Results show in Figure 5-3 indicate that according to the logged schedules in project development, around 450 man-hours were cost in design, implementation and training of the framework. The cost of Game 3, 4, 5 and 6, using framework, were calculated as 70, 80, 120 and 100 man-hours respectively, which included scene building, GameObject hierarchy building, user interface design, game mechanic programming and parameter configuration. In contrast, the implementation of Game 1 and Game 2, which were built from scratch without the framework, took 320 and 240 man-hours respectively. Our experience in building VR exergames (Ijaz et al., 2016a, 2016b) without a framework show that approximately 25% of effort cost in game development could be saved by using the existing implementation of first game (Game 1), such as hardware connection and parameter configuration from the existing code. As can be seen in Figure 5-3, the cost comparison of different games indicates that when a set of VR exergames that use the same hardware platform are required for different user studies, it is justified to develop a framework for game implementation.
5.4 Framework code quality analysis

Besides the evaluation of metrics on framework efficiency in assisting the development of VR exergames, the software metrics on the reusable code provided by framework such as design complexity metric, maintainability metric and reusability metric also need to be considered. The results of code analysis provide a better understanding of the code’s reliability and maintainability, which in turn allow the developer to identify potential risks and test applications properly so that improving software engineering from managerial and technical perspective (Fenton & Neil, 1999).

Therefore, this study analysed the quality of the framework’s source code with standard code metrics data (Microsoft, 2016), which were calculated by the standard Code Analysis function tool in Microsoft Visual Studio 2015. The code metrics used in this software measurement were Maintainability Index, Cyclomatic Complexity, Depth of Inheritance and Class Coupling. The boxplots in the following subchapters show the framework code metrics data analysis result distributions of all the modules’ source code classes in this framework.

5.4.1 Maintainability index

This index was first introduced by Oman et al. (1991) as a composite metric for measuring software maintainability. Been successfully used for decades in various industrial systems, this index has provided valuable insights in software maintainability issues (Welker, 2001). It is an incorporating of several traditional source code metrics (Oman & Hagemeister, 1992) such as weighted Halstead metrics (Halstead, 1977), McCabe’s Cyclomatic Complexity (McCabe,
1976) and lines of code (LOC). In brief, this index calculates the level of code maintainability, which is an effective approach commonly used in quantifying the software maintenance.

In Microsoft Visual Studio, maintainability is calculated with a result index value ranging from 0 to 100. According to the specification in Microsoft Visual Studio Documentation, Visual Studio re-sets the calculation results to lie between 0 and 100, instead of the result ranging from 171 to an unbounded negative number. This makes the results clearer to understand since the code with original result less than 0 is hard to maintain.

The results are rated in three levels by Visual Studio. For instance, value between 20 and 100 is a green rating (value is showing with green colour) which indicates a good maintainability for the code; while value ranging from 10-19 represents moderately maintainability, showing with yellow colour; and value ranging from 0-9 represents low maintainability, showing with red colour. The calculation results of Maintainability Index of the framework code are illustrated in Figure 5-4.
CHAPTER 5. FRAMEWORK CODE EVALUATION

Figure 5-4 shows the distribution of the framework code Maintainability Index calculation results in boxplot representation. This boxplot depicts that the median of the Maintainability Index on the framework code is 72. This value is reasonably good, as it is significantly above 20 (the minimum green rating value suggested by Microsoft Visual Studio). The results commonly belong to a range between 67 (Q1) and 81 (Q3), which means 50% of the results are within this range, indicating a high maintainability. The lowest value is 54, which is still significantly above the floor value of 20, which is the minimum value for good maintainability. Based on the Maintainability Index calculation results, it can be summarized that the framework code is relatively easy to maintain.

5.4.2 Cyclomatic complexity

Developed by McCabe (1976), Cyclomatic Complexity is a measurement of software complexity based on graph-theoretic concept. This index focuses on the decision structure of a software program, which is independent from the functions in classes. To be specific, Cyclomatic Complexity calculates the amount of decision logic made in every source code function to measure the structural complexity. In other words, it computes the number of linearly independent circuits in the control flow graph of a software’s source code, where three variables in the graph - the number of edges, number of nodes and number of connected components – are used for calculation. Furthermore, it is a mathematical method which provides a quantitative knowledge for identifying the difficulty in test and maintenance of a software, module, function or class (Schroeder, 1999).
The Cyclomatic Complexity calculation results of all the framework modules’ classes are shown in Figure 5-5. According to Microsoft’s documentation, this index in Visual Studio indicates the difficulty level of a code to test and maintain. A high value indicates less maintainable code compared with a low value, because it requires more tests for a better code coverage. On the other hand, based on the structured testing from Watson et al. (1996), an upper limit of 10 is an optimal value for common projects. The distribution of this index in Figure 5-5 reveals that the results are commonly located in the range between 1 (Q1) and 2 (Q3), which means that the Cyclomatic Complexity of most of the framework’s source code classes are 1 or 2. Although there are some outlier values which mean that there are very few classes’ index between 4 and 9, all of these outlier values are less than the optimal value 10. This boxplot reveals a good distribution of the calculation results, from which it can be concluded that the framework code’s classes structural complexity is low, indicating high maintainability and low test difficulty.
5.4.3 Depth of inheritance

Proposed by Chidamber & Kemerer (1994), Depth of Inheritance is a metrics for measuring object-oriented software designs. This index is defined as “depth of the class in the inheritance tree”, in which the inheritance tree is a mechanism in object-oriented programming that allows the subclass to reuse and extend the code from parent class. This metrics not only provides developers a mathematical method for quantifying the complexity in object-oriented design, but also helps in reusing code, easing the testing stage, and determining software defects for designing high-quality software (Subramanyam & Krishnan, 2003).

In this object-oriented design method, deep inheritance tree indicates greater complexity because more approaches and classes are included. According to Chidamber & Kemerer, a higher value of depth implies more complexity in the prediction of the class’s behaviour so that it is less maintainable. However, in Microsoft Visual Studio Documentation also introduces

![Depth of Inheritance analysis results of framework code](image-url)

**Figure 5-6. Depth of Inheritance analysis results of framework code**
that higher number of depth implies more potential for code reuse through inheritance. Defined as “indicate the number of class definitions that extend to the root of the class hierarchy” in Visual Studio documentation, this code analysis function in Visual Studio will issue a warning (excessive inheritance) when this index reaches 6 or greater, based on Microsoft’s specification, 6 is an upper limit.

Figure 5-6 illustrates interesting results from Depth of Inheritance calculations for all the framework modules’ classes. On one hand, the overall results are below 6, which indicates that the results are within reasonable range and excessive inheritance is avoided. On the other hand, the overall results are commonly vary between 3 to 5, meanwhile the distribution is skewed right where most values are 5. This implies that the framework modules have great potential for code reuse.

5.4.4 Class coupling

Class Coupling was first introduced with Depth of Inheritance by Chidamber & Kemerer (1994) as a set of metrics for improving object-oriented design. Essentially, Class Coupling calculates the number of classes that a single class uses. Low value is generally considered good and high number is bad with this metrics because high coupling indicate difficulty in reuse and maintain of the design, where it depends on too many other types (Kabaili et al., 2000). The quantitative investigation in object-oriented metrics (Shatnawi, 2010) has proved that Class Coupling can be used as a precise predicator of software failure. Shatnawi also show that an upper-limit value of 9 is optimal for efficiency.
Visual Studio calculates this index through parameters, local variables, return types, etc. The Class Coupling calculation distributions of all the framework modules are shown in Figure 5-7. According to this figure, the median of the results is 1 and most of the results are in a range between 0 (Q1) and 3 (Q3), which indicate that the majority of the framework class designs has a low coupling. The maximum value in this distribution is 7 that is still below the upper-limit value 9 implying a relatively low coupling. There are also very few outliers at 8 and 9 but they remain within the optimal limit. From Figure 5-7 it can be summarized that the framework code is generally easy to reuse and maintain because of the low Class Coupling.

5.5 Discussion

In this framework code evaluation, the immersive VR exergames that were built by our lab for conducting user experience studies were compared to evaluate the performance of this framework in software development. This evaluation is important because the framework is
designed to be used by other developers and researchers in HCI community. Among the exergames used for comparison, two were built without using any framework and four were built using the framework proposed in this thesis.

The evaluation consists of two parts, the framework code efficiency study and code quality study. In detail, the code efficiency study compares the amount of effort that the framework code helps to reduce in each VR exergame, which includes the amount of code provided by framework and the time cost in each game. Ideally both efforts in code writing and time consuming could be significantly reduced with framework. The results of this study showed that by using the modules of this framework in VR exergames development for the particular functionalities, more than 70% of effort were saved in code writing. In addition, it can be observed from the cost comparison between exergames that this framework benefits an HCI research team when there are different VR exergames required for experiments.

The reasons leading to the results above are that the framework provides not only the reusable code in each module for easing the integration of VR devices and exercise related devices into Unity projects, but also the reuse of practices in experiment-oriented exergaming domain which developers can easily add exercise experiences such as staging, physiological data to collection and user study grouping and recording. Modules can be singly used for one functionality or combined for multiple features. The use of framework modules barely requires any code writing work from developers, so both the reuse methods reduce the time and effort for building immersive VR exergames for user studies.
The code quality study calculates the standard index of the code quality to evaluate the overall quality (such as maintainability and reliability) of the framework. Results suggested that the framework modules’ source code have a low complexity, indicating that it is easy to understand and less tests are required to achieve good code coverage. The source code also has relatively low class coupling and medium depth of inheritance, that also imply good potential for code reuse. The maintainability index is quite high which means the source code has a good maintainability. Therefore, this framework brings the features such as ease to test, good reliability and maintainability to the VR exergames built with it.

Given the above results, it can be concluded that the framework can help to reduce the investment of time and effort in the implementation of VR exergames. Meanwhile the framework itself is reliable and maintainable, which also contributes to a good code quality in the VR exergames built with it.

**Limitation**

There are several limitations to this code evaluation that should be considered. The main limitation was that the number of VR exergames not using framework and those using framework in the comparison study were limited to two and four. This was because that these applications were built for conducting different ethics approved user studies, where the applications for each human ethics protocol took few months. Due to the fact that each game had different designs and not all the modules were used in every exergame built with framework, the code efficiency study was limited to compare the difference in SLOC and IL
code provided by framework modules in each game. In addition, the code metrics in object-oriented programming are not limited to the four analysed in the thesis, there are similar metrics for code evaluation such as “Lack of Cohesion Of Methods” and “Weighted Methods Per Class” (Chidamber & Kemerer, 1994). This thesis used the standard tool in Microsoft Visual Studio 2015 to calculate these four metrics.

**Conclusion**

In conclusion, although there were only few samples of applications that were built with this framework, the code evaluation provided us an overall understanding of how this framework helps developers. The effort investment in code writing and time can be significantly reduced with this framework. Importantly, this chapter provided the evidence that the framework code is reliable and easy to maintain, which contributes to a good code quality of the applications built with it. The good code metrics results also ease the future work in extending the framework functionalities.

Overall, this chapter demonstrated that this framework is promising as a tool for HCI community to employ in building experiment-oriented VR exergames. It is still necessary to build more VR exergames with this framework for a more comprehensive code efficiency evaluation in the future. A code quality analysis that includes more code metrics can also be taken into consideration for a thorough understanding of framework quality in future work.
Chapter 6. Framework Feasibility and Usability Evaluation

Summary

Besides the evaluation of the framework from a software engineering perspective, it is also necessary to assess it from an HCI perspective to evaluate how the framework fits its objective – producing different VR exergames for HCI user studies. Therefore, this chapter investigates the framework’s feasibility and usability through the data collected in user studies. The results of feasibility study show that this framework can promotes players to do moderate exercise without motion sickness, in which the success rate was significantly high. The outcomes of usability evaluation suggest that VR exergames built with this framework engage the participants. These two evaluations analyse the data collected from our previous user studies published as (Ijaz, Ahmadpour, et al., 2020; Ijaz et al., 2017, 2016a, 2016b; Ijaz, Wang, et al., 2019) in the Preface section, where I am the second author. All the contents in this chapter, which are related to these papers, are my original work as a co-author and have been permitted by the corresponding author to be used in my thesis.
6.1 Introduction

The code evaluation in Chapter 5 shows that this framework helps to reduce efforts such as code written, time spent and hardware integration in application development. Meanwhile the framework code itself has a high maintainability and reliability, as well as great potential in code reuse. However, this evaluation is from the perspective of software engineering, which only covers the framework’s performance in development stage and framework code quality. It does not provide a thorough investigation of a tool that is designed and developed for others to use, e.g. HCI research community.

In order to further explore how the framework performs in HCI studies, this chapter uses part of the data collected in previous user studies for analysis. All the studies were conducted with the VR exergames mentioned in Chapter 5, which were built using same hardware setup and framework modules. Though the user studies were designed to investigate the motivations to exercise with VR, some of the data collected can be used to evaluate the framework. This chapter reports the data analysis’s findings correlated to the framework setup, usage and performance in those projects. The use of partial data in my thesis has been permitted by the corresponding author.

User study details on participants information and method used in the studies are introduced in subchapter 6.2. Data analysis on feasibility evaluation is discussed in subchapter 6.3. Framework usability in terms of participants’ engagement is evaluated in subchapter 6.4.
6.2 User studies

Three user studies have been conducted using our VR exergames, one pilot study and two research studies, which were approved by The University of Sydney Human Research Ethics Committee (HREC) with protocol number 2015/185 and 2016/996. The pilot study was a preliminary study to test the feasibility of hardware setup and virtual environment, in which the exergames were built from scratch without using any framework. The two research studies were motivation studies on exercise with virtual reality, with all the exergames developed using our framework modules.

The steps of our user studies in HCI domain include: 1) researcher recruits participants from university or local community centre through flyers; 2) researcher introduces participant information statement to registered participant and gets participant’s consent; 3) participant fills out screening pre-test questionnaire; 4) researcher assists participant to adjust VR setup; 5) participant plays VR exergames in different study conditions (between-subjects or within-subjects according to each study); 6) participant fills out screening post-test questionnaire. Besides, participants can withdraw from the study at any time. The details of each user study are introduced below.

6.2.1 User study 1

This was a pilot study to test the feasibility of our hardware setup and virtual environment. It employed Game 1 and Game 2 introduced in Chapter 5 (both built without using any framework) to investigate how different game designs impact user experiences. Though the
game mechanic designs were totally different, the two games used a same hardware setup, including recumbent trike, speed sensor, mobile phone, VR headset and VR hand controllers, allowing players to freely cycle in the same virtual environment and interact with the scenes in different ways using controllers.

- **Participants**

In this pilot study, younger undergraduate participants from our university and older participants (all aged over 60) from a local community centre were recruited respectively over a period of three months. Ninety-two participants aging between 18 to 96 years registered for this study. One younger participant was excluded due to technical issue and one older participant had to be excluded due to experiencing discomfort with VR headset. The remaining 90 participants (40 males and 50 females, M = 45.66, SD = 23.63) were randomly allocated to either competitive group (N = 45) or affiliative group (N = 45), where the number of younger participants and older participants were approximately same in each game condition.

- **Method**

In this study, each participant was allocated a time slot of 30 minutes to play either Game 1 (competitive guessing game) or Game 2 (affiliative tour game) until game finished. Our data were collected through the screening pre-test and post-test questionnaires with five-point Likert scale (Likert, 1932), including various questions about personality preferences, task enjoyment, motivation, and preference for future play. The questionnaires (see Appendix 1) are based on the Revised Competitiveness Index (Houston et al., 2002) that proves to internally consistent
(Harris & Houston, 2010). All the questions were rated from 1 (do not agree) to 5 (strongly agree) by participants. During the gameplay, participants’ verbal reactions were also recorded as audio files using think-aloud protocol (Ericsson & Simon, 1980), as well as the gameplay data recorded into XML files.

6.2.2 User study 2

This study used Game 3 and Game 4 introduced in Chapter 5, which were built using framework modules, to explore the motivation and enjoyment of physical activity on different VR exergaming designs. These two games shared same hardware setup, but totally different game designs and virtual environment. Game 3 was a simple game without panoramic view (UI, condition 1) and Game 4 was an explorable open space (OpenWorld, condition 2). The hardware setup involved wristband, trike, speed sensor and HTC Vive, allowing participants to freely cycle in a virtual world and interact with scenes using controllers.

- Participants

In this study, participants were all recruited from our university through flyers. Eighty-nine participants registered over a period of two months (two excluded because of technical issues). The remaining 87 participants were students and staffs aged 17 to 59 (M = 24, SD = 2.12, Male = 60, Female = 27) and they were randomly allocated to one of two experiment conditions (43 in condition 1 and 44 in condition 2).

- Method

In this study, participants needed to complete a pre-test questionnaire including basic
information (age, gender, weight, height), exercise routine (weekly physical activity frequency),
video game routine (weekly gameplay frequency) and prior VR experience, etc. They were
required to wear a wristband for physiological data collection during gameplay. The game
conditions were randomly allocated (either condition 1 or condition 2), and they were left to
play until they called to stop. Post-test questionnaire included Physical Activity Enjoyment
Scale (PACES, see Appendix 2) (Kendzierski & DeCarlo, 1991), Player Experience of Need
Satisfaction (PENS) (Ryan et al., 2006), future play intension and other experiences. In our
questionnaires, PACES used 7-point Likert questions to evaluate vitality and enjoyment of
physical activity. PENS also used different 7-point Likert questions to address intrinsic and
extrinsic motivations for acting, which was elaborated from self-determination theory (SDT)
(Ryan & Deci, 2000). In addition, the verbal reactions were recorded as audio files and the
gameplay data were recorded as XML files.

6.2.3 User study 3

This is an ongoing study by the time the thesis is being written. This study used Game 4
(modified version, new locations and new panoramic views were applied, DeskCycle was used
to replace the trike) and Game 5 introduced in Chapter 5. The two games were built using all
framework modules to analyse the motivation and enjoyment of physical activity on different
VR exergaming designs. Pre-test and post-test questionnaires same as user study 2 were used,
in which basic demographic data, physical activity routine data, standard questionnaires for
motivation research were collected.
Participants

In this study, 30 student participants aging from 19 to 34 (M = 25, SD = 3, Male = 15, Female = 15) have been recruited from our university. Each participant played all the three game conditions.

Method

Though the study steps were same as user study 1 and 2, each participant in this study was required to play all three games – firstly play condition 1, secondly play either condition 2 or 3 (randomly allocated), thirdly play the rest condition. The three game conditions were: 1) condition 1 - open world for free exploration, i.e. Game 4 without any gameplay status (time, distance, physiological data, in-game map); 2) condition 2 - open world with gameplay data, i.e. Game 4 with all gameplay status (time, distance, physiological data, in-game map); 3) condition 3 - open world with real-world video game elements, i.e. Game 5 (a cycling and shooting game). After the gameplay session, there was a short interview section before filling out the post-test questionnaire, where several questions about game experience were asked and verbally answered. All the game data were recorded as XML files and all the verbal reactions and interview answers were recorded as audio files.

6.3 Framework feasibility evaluation

The feasibility study of a framework is a study that measures whether the proposed framework solves the problems and how it satisfies the requirements (Bentley & Whitten, 2007). In our case, the framework is designed and developed for HCI community to create VR exergames,
where proper physical activity and low motion sickness are required. Thus, this study evaluates the feasibility of framework through the applications used in our previous user studies, which employed same hardware setup and were built using framework modules.

The evaluation is based on our previous collected data in user studies, in which four aspects from feasibility are analysed, namely success rate, exercise performance, motion sickness and perceived competence respectively.

### 6.3.1 Success rate

The success rate analyses the results of participants succeeded in finishing the gameplay sessions in each study. This factor is important for user studies which recruit participants to use our applications and provide feedbacks, because it directly reflects the stability of our VR exergames in studies. Therefore, this evaluation calculates the success rate in each study. Though the games in user study 1 were not built with framework, these two exergames shared the same hardware setup with the games in user study 2, which can be used to assess the framework module’s hardware feasibility. The data are shown in Table 6-1.

According to Table 6-1, the number of participants registered for user study 1 is ninety-two.

<table>
<thead>
<tr>
<th></th>
<th>Number of registered</th>
<th>Number of succeeded</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>User study 1</td>
<td>92</td>
<td>90</td>
<td>97.83%</td>
</tr>
<tr>
<td>User study 2</td>
<td>89</td>
<td>87</td>
<td>97.75%</td>
</tr>
<tr>
<td>User study 3</td>
<td>30</td>
<td>30</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 6-1. Success rate of participants in user studies
Two of them did not finish the study due to technical issue and discomfort with VR headset. All the rest 90 participants finished the game session and provided feedbacks through pre-test and post-test questionnaires. In this case, we have 97.83% success rate in user study 1, indicating that our hardware setup with recumbent trike, sensor, and HTC Vive (headset and hand controllers) is significantly successful in human study where both young students and old adults can do proper exercise with it. Furthermore, the number of old participants aging above 60 years old meanwhile successfully finished the study is 36. It is important to point out that there were 9 participants aging above 80 and two of them were 92 and 96. This depicts that our hardware setup is very safe and friendly to old adults.

In user study 2 where all the participants were university students and staffs, we have a success rate as 97.75% in 89 registered participants. Though two of them experienced technical issue where we had to abandon the data, the success rate is considerably high. In user study 3, we have 100% success rate in 30 registered participants. Even though the number of participants is much less than user study 1 and 2, the success rate still reasonably proves that this framework is reliable to produce applications for user studies.

Based on the fact that the overall success rate is 98.1% in the three studies which recruited 211 participants in total, we can argue that this framework’s hardware setup is feasible for building VR exergames for user studies, especially it is senior friendly. Though technical issues occurred, the high success rates in user study 2 and user study 3 still demonstrate that the applications built with framework are reliable in practice.
6.3.2 Exercise performance

The exercise performance evaluates whether the framework can promote physical activity through exergames developed with it. In this evaluation, exercise data in terms of activity time are analysed. The activity time in each gameplay session are shown in Table 6-2.

In user study 1 there were 45 participants spending average 6:27 mins (SD = 2:30) in game condition 1 (competitive game), while another 45 participants played game condition 2 (affiliative game) for average 9:58 mins (SD = 4:16). These are reasonably good results for a pilot study, especially the average gameplay time for each participant in condition 2 is approximate 10 minutes. The longest gameplay time is 19:21 mins, which is a long time for cycling. The results indicate that the hardware setup is successful in engaging participants to do exercise.

In user study 2, forty-three participants spent average 4:25 mins (SD = 2:36) in game

<p>| Table 6-2. Activity time in each study (min:sec) |</p>
<table>
<thead>
<tr>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User study 1</strong></td>
<td>1:49</td>
<td>11:50</td>
<td>6:27</td>
</tr>
<tr>
<td>Condition 1 (N=45)</td>
<td>1:10</td>
<td>19:21</td>
<td>9:58</td>
</tr>
<tr>
<td>Condition 2 (N=45)</td>
<td>1:07</td>
<td>12:03</td>
<td>4:25</td>
</tr>
<tr>
<td><strong>User study 2</strong></td>
<td>2:09</td>
<td>27:20</td>
<td>7:54</td>
</tr>
<tr>
<td>Condition 1 (N=43)</td>
<td>4:25</td>
<td>26:49</td>
<td>11:43</td>
</tr>
<tr>
<td>Condition 2 (N=44)</td>
<td>4:43</td>
<td>18:20</td>
<td>10:21</td>
</tr>
<tr>
<td><strong>User study 3</strong></td>
<td>4:05</td>
<td>25:52</td>
<td>16:09</td>
</tr>
<tr>
<td>Condition 1 (N=30)</td>
<td>4:25</td>
<td>26:49</td>
<td>11:43</td>
</tr>
<tr>
<td>Condition 2 (N=30)</td>
<td>4:43</td>
<td>18:20</td>
<td>10:21</td>
</tr>
<tr>
<td>Condition 3 (N=30)</td>
<td>4:05</td>
<td>25:52</td>
<td>16:09</td>
</tr>
</tbody>
</table>
condition 1 (Game 3), while 44 participants played game condition 2 (Game 4) for average 7:54 mins (SD = 4:57). The activity time in condition 2 is longer than the time in condition 1, due to the reason that Game 4 has an explorable virtual environment and the scene in Game 3 cannot be explored.

In user study 3 where each of 30 participants played all three conditions, the average cycling time in each condition is longer than 10 minutes, suggesting that the three games built with all the framework modules can engage participants to do moderate cycling exercise. In addition, the minimum, maximum and average activity time of each condition in user study 3 are substantially longer than those in user study 2. This is because that the framework has been updated and optimized for supporting diverse cycling hardware and better interaction, which attract participants to spend more time in our VR exergames.

In total, from the data in Table 6-2 we can summarize that the hardware setup used by this framework (trike/DeskCycle, VR device, sensors) fits both young and old participants to do moderate cycling activity. The virtual scene and interaction supported by the framework modules in user study 2 and 3 engage players to spend more time playing the games. The exergames developed with framework have an overall good exercise performance for players, indicating that this framework is feasible for HCI studies.

### 6.3.3 Motion sickness

Motion sickness is an important aspect of a VR system as it is directly related to the feasibility. Motion sickness is produced by conflicts between inputs (such as visual and acoustical inputs)
and responses in humans (Ohyama et al., 2007), which is a morphological and physiological connection issue (Money, 1970). Though the virtual scene in VR system may cause motion sickness, a proper design of virtual environment and appropriate interactive method (mapping hardware interaction to software response) can increase the fidelity in VR to reduce the motion sickness for users. In this evaluation, we analyse participants’ reactions in terms of motion sickness when playing our VR exergames to study the feasibility of the framework. Data are presented in Table 6-3.

The motion sickness section was not included in the questionnaire of our user study 1, so the data are from the study 2 and study 3. According to the table, in the pre-test questionnaire of user study 2, seventeen participants indicated likelihood of motion sickness on a moving vehicle, which is 19% of the 87 participants. Results from the post-test survey show that 8 participants (9%) experienced motion sickness in gameplay session. It is worth to point out that nearly same number of participants in both condition 1 (N = 9) and condition 2 (N = 8) of this study reported experiencing motion sickness, where the game in condition 1 did not have navigation or movement in virtual space that could cause sickness.

<table>
<thead>
<tr>
<th></th>
<th>Likely to have motion sickness</th>
<th>Experienced motion sickness</th>
</tr>
</thead>
<tbody>
<tr>
<td>User study 2 (N=87)</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>User study 3 (N=30)</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6-3. Number of participants that were likely to have motion sickness and those experienced motion sickness in user studies
In user study 3, ten participants (33% of 30) indicated possibility of motion sickness on moving vehicle, whereas the result of post-test survey shows that 2 participants (6%) experienced motion sickness in one gameplay of all three conditions.

Total number of 10 (8%) out of 117 participants got motion sickness playing the VR exergames that were developed with this framework. The results demonstrate that our framework is capable of reducing motion sickness in the applications built with it.

### 6.3.4 Perceived competence

Studies have suggested that perceived competence contributes to both physical activity and gaming experience (Amold, 1985; Fairclough, 2003; Williams & Gill, 1995). In our feasibility study of the framework, perceived competence refers to the capability and ability of a participant to understand to control the environment, outcome and experience (Deci & Ryan, 2004). In a VR exergame context, perceived competence is a self-perception of an individual on how skilled and effective to be in the VR exergaming situation. In our study, perceived competence indicates the difficulty level for a player to master to play the exergames built with framework, i.e. the game is easy or hard to understand, or easy or hard to adjust the player himself during gameplay.

Our data were collected through standard measurement in post-test questionnaire, where participants were required to rate their level of agreement on different questions that were equally in scoring about perceived competence such as “I felt very capable and effective”. The survey results of 5-point Likert scale (user study 1) and 7-point Likert scale data (user study 2
and 3) on perceived competence (median value of responses) are presented in the following table.

Table 6-4 shows the results from all three user studies. From the table we can see that more than half of the participants (N = 51) in user study 1 rated neutral on perceived competence, indicating that it was neither easy nor hard for them to control their game experience. In particular, 28 participants (31%) rated agree, revealing that they found it easier to master the skills in playing these VR exergames. Although none rated strongly agrees, the overall results show that the participants were capable of controlling the game outcomes.

In user study 2, a total number of 18 (21%) participants rated difficult to master their game experience, whereas the rest 69 participants (79%) rated neutral or easy to master the skill. Nineteen (22%) participants rated strongly agree, showing that they were completely able to adapt themselves during gameplay for better gaming results and performance. The results of
user study 3 are interesting, in which none of the participants (N = 30) rated disagree. Nearly half of the participants (N = 14) felt easy to know how to play the games during all the three gameplay. One of the reasons is that the framework modules had been updated to support more hardware devices (e.g. DeskCycle) and interaction methods since user study 2, which improved user’s perceived competence in the games developed with framework modules.

The results of analysis demonstrate that the framework is feasible for creating VR exergames for user studies. On one hand, the participants felt easy to control their game outcomes in our studies, indicating the interaction through cycling and hand controllers are easy to master for gaming performance. On the other hand, the overall perceived competence results in user study 3 are better that that in user study 2, indicating that the improvements on supporting diverse hardware devices in software modules are successful to promote the games developed with framework to achieve better in-game competence.

6.4 Framework usability evaluation

Since our framework targets to support HCI researchers to build VR exergames for user studies, the evaluation of framework should include not only the software engineering code analysis and feasibility analysis but also the usability analysis on how it satisfies the end-users, i.e. the participants in studies. Usability study in software engineering refers to analysing the quality of a user’s reactions and overall satisfaction on a software application or system (Kirakowski, 1996). In our case, the usability evaluation of the framework is a study that measures how the participants are satisfied and engaged with the exergames built with framework.
This evaluation is based on our previous collected data in user studies. We analyse the usability through a combination of factors including enjoyment, immersion, intuitive controls, preference for future play and user observation.

6.4.1 Enjoyment

Enjoyment is defined by Davis (1982) as “feeling that causes a person to experience pleasure”, where pleasure is occasional happiness and excitement. HCI researchers have emphasised the significance of enjoyment in designs, which contributes to usability and user experience (Feng et al., 2008; Malone, 1982; Sweetser & Wyeth, 2005). Consequently, enjoyment has been one of the goals and included in usability concepts in software design by HCI research community (Monk et al., 2002). Besides, considering enjoyment in software design is an important factor that improves product development and leads to product success (Marc Hassenzahl et al., 2001).

The typical measurements of enjoyment in HCI studies are through self-reports (Bateman et al., 2010; Li & Moacdieh, 2014) or questionnaires (Saket et al., 2016) after participants performing the tasks. Therefore, our studies used standard surveys (PACES and PENS), with different questions such as “I enjoyed playing the game very much”, to collect the enjoyment results (median value of responses) listed in the table.

According to Table 6-5, the enjoyment feedbacks in user study 1 were rated with 5-point Likert scale and the results are of great significance. None of the 90 participants rated “do not agree” during the gameplay sessions in this study, while 8 participants (9%) rated neutral. Eighty participants (90%) rated agree and strongly agree, indicating they enjoyed playing the
games. The results depict that the virtual environments and the interaction methods designed for these two exergames are successful for providing enjoyment in user study.

In study 2, the degree of agreement on enjoyment from 87 participants distribute evenly. A total number of 25 participants (29%) rated agree and strongly agree. Meanwhile the perceived enjoyment degree of 46 participants (53%) were around neutral. It is worth mentioning that 14 participants in condition 1 (Game 3) rated disagree, where this game did not have 3D scenes and the virtual environment could not be explored. The results of user study 3 show that 47% of the participants (N=14) perceived pleasure during all the gameplays, indicating that the updated framework modules performed well in creating VR exergames that are engaging.

From the enjoyment analysis of the VR exergames employed in user studies, it can be summarized that both the hardware setup of framework modules and software designs (virtual scenes, exercise data, maps, interactions, etc. brought by the framework) contribute to the enjoyment of players. The results of enjoyment across all three studies with 207 participants

Table 6-5. Frequency of participants’ responses on the agreement on enjoyment

<table>
<thead>
<tr>
<th></th>
<th>Do not agree</th>
<th>Neutral</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>User study 1 (N=90)</td>
<td>0</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>User study 2 (N=87)</td>
<td>5</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>User study 3 (N=30)</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
are generally good, which shows the framework’s great usability in creating experiment-oriented VR exergames since a high level satisfaction on enjoyment are perceived by players.

6.4.2 Immersion

In gaming environment, immersion is a sense of presence based on gamer’s experiences, which is used for representing the degree of participation (engagement, engrossment and total immersion) in a game (E. Brown & Cairns, 2004). In virtual reality, immersion refers to the perception of physically presenting in a digital world, which provides benefits in different VR contexts (Bowman & McMaban, 2007). As an important gaming factor, immersion is widely measured through standard questionnaires in research studies involving computer game narratives (Brockmyer et al., 2009; Qin et al., 2009). Similarly, as the most important feature of VR, it is necessary to evaluate the immersion of a VR system. Therefore in our case, we investigate the perceived immersion brought by our framework modules through PENS questionnaire (Ryan et al., 2006), which assesses physical presence, emotional presence and narrative presence in the gaming environment. In user study 1, PENS questionnaire was not used. The overall results (median value) of user study 2 and 3 are shown in Table 6-6.

| Table 6-6. Frequency of participants’ responses on the agreement on immersion |
|------------------|--------|--------|--------|--------|--------|
|                  | Do not agree | Neutral | Strongly agree |
| User study 2 (N=87) | 22    | 16    | 13    | 14    | 9      | 10    | 3     |
| User study 3 (N=30)  | 0     | 2     | 3     | 8     | 5      | 11    | 1     |
According to this table, the perceived immersion of 38 participants (44%) were around disagree in user study 2, while only 13 participants (15%) rated agree and strongly agree. Among these participants, there were 25 allocated to play condition 1 (Game 3) who rated disagree, taking up 29% of the total participants. We have pointed out that Game 3 did not contain any explorable environment. Instead, it was a static UI in a virtual scene (although the player still had a 360-degree view). Despite this simple design significantly reduced the presence for players, we intentionally designed and developed it in this way to study the motivation for exercising in different VR environments. Nevertheless, from the results of questionnaire we can still conclude that the overall immersion brought by the framework modules are good.

In user study 3, only two participants reported disagree, and the rest participants (93%) reported around neutral (53%) and agree (40%). The results of study 3 have noticeably shown that the three exergames developed using all the framework modules deeply immersed the players, demonstrating the usability of our framework in terms of immersion meets the requirement of VR studies.

In summary, the overall results of immersion analysis imply that our framework assists to create VR exergames that supports emotionally engaging on perceived immersion. In this way, our framework modules help to improve the presence experience for better study results, indicating that our framework has great usability on immersion.
6.4.3 Intuitive controls

Intuitive controls represent the interface that controls the player’s actions in the game environment. It is related to player’s sense of freedom, control and competence that contribute to game motivation (Ryan et al., 2006). Therefore, intuitive controls can enhance player’s experiences such as enjoyment during gameplay. This thesis uses PENS questionnaire to assess the intuitive controls of the games in previous studies, which include several questions such as “When I wanted to do something in the game it was easy to remember the corresponding control.” Questions were rated with 7-point Likert scale and the results (median value of responses) are presented in Table 6-7. Since PENS was not used for user study 1, this analysis only includes user study 2 and 3.

Based on Table 6-7, only two participants (2%) in user study 2 rated the game controls not intuitive (do not agree), meaning that only very few participants felt the game controls hard to master. While among the rest 98% of the participants, 61 (70%) of them thought the game controls were intuitive (agree and strongly agree), revealing that the controls were easily mastered and their senses of being in the game were not interfered at all.

| Table 6-7. Frequency of participants’ responses on the agreement on intuitive controls |
|---------------------------------|-----------------|-----------------|-----------------|
|                                 | Do not agree    | Neutral         | Strongly agree  |
| User study 2 (N=87)            | 1               | 1               | 4               |
|                                 | 8               | 12              | 26              |
|                                 | 35              |                 |                 |
| User study 3 (N=30)            | 0               | 0               | 2               |
|                                 | 4               | 6               | 11              |
|                                 | 7               |                 |                 |
In user study 3, none of the participants rated “do not agree”, suggesting that all of them thought the game interfaces (brought by framework modules) were user-friendly. Besides, 18 participants (60%) indicated that the game interface design merely controlled their in-game actions by rating the game controls intuitive (agree and strongly agree).

The overall positive feedbacks on intuitive controls from the user studies outline that the VR exergames developed with our framework facilitate player’s gaming autonomy experience. On the other side, the intuitive design brought by framework allows players to put efforts on game play instead of game mechanics. It can be concluded that the framework modules also have great usability on intuitive controls.

6.4.4 Preference for future play

Preference for future play reveals a player’s overall satisfaction of a game. Intended future play is significantly related to player’s motivation and enjoyment outcomes as these factors contribute to future play (Yee, 2006). The preference for future play in our user studies are presented in the following Table 6-8.

According to the table, 73 participants (81%) in user study 1 would like to play the VR

<table>
<thead>
<tr>
<th></th>
<th>Willing to play in future</th>
<th>Unwilling to play in future</th>
</tr>
</thead>
<tbody>
<tr>
<td>User study 1</td>
<td>73</td>
<td>17</td>
</tr>
<tr>
<td>User study 2</td>
<td>57</td>
<td>30</td>
</tr>
<tr>
<td>User study 3</td>
<td>29</td>
<td>1</td>
</tr>
</tbody>
</table>
exergames in future sessions, suggesting that our framework’s hardware setup and the cycling game design (mapping cycling to in-game movement based on pedalling speed) in VR were user-friendly and attractive, especially for senior players. Though 30 participants (34%) in user study 2 were unwilling to play in future. This is still reasonable because 19 of them played the UI condition which was less attractive. The rest 66% participants thought the games worth to play in future, revealing that the general system setup and game mechanics brought by framework modules were acceptable for the majority of participants. In user study 3, only 1 participant (3%) reported to not to play again in future, which means that all the three exergames attracted and satisfied participants. In total, the feedbacks from all three studies on preference for future play have outlined that the framework has good usability in creating VR exergames that players would like to play multiple times.

6.4.5 Observation

To provide further insights into participants’ experiences and satisfaction on the VR exergames built with framework, their verbal feedbacks were collected using think-aloud technique.

In user study 1, participants were immersed in the virtual environment, stating “It feels real as in the traffic.”, “I can get so close to the landmark.”, “Do I need to wait for traffic lights?”, “This is the future, O man!” etc. More importantly, the games recalled some senior players’ memories. “I used to jog on this street.”, “I have been here twenty years ago.”, said the seniors. The responses from the participants were positive, which reveal the virtual scene attracted players and improved their gameplay experiences.
Participants in user study 2 and user study 3 also provided positive feedbacks in terms of enjoyment such as “You can turn with trike, aw pretty cool.”, “This is actually quite nice, the experience.”, “When I play this game, time goes faster.”, “I’m not a game person but I’m bit addicted to it now”. There were also statements on immersion, e.g. “Wow there’s car coming.”, “I’m trying my way home.”, “Opera House is coming.”, showing a good presence experience brought by our virtual scene. Further, there were also comments revealing the intuitive game controls such as “I can just cycle to museum, excited.”, “Yes, pedalling fast makes me faster.”, “Good that I can check my location on the map.”.

Since all our VR exergames share the same hardware setup, virtual environment and cycling interaction that are integrated in framework modules, the participants’ positive verbal feedbacks on enjoyment, immersion, and intuitive controls are reliable predictors of the good usability of our framework.

6.5 Discussion

In this framework feasibility and usability evaluation, the data collected from our user studies were analysed to reveal the performance of exergames built with framework. This evaluation is necessary because the framework is designed for user studies in HCI research community.

In the three studies with data collected, the games in user study 1 were not built with framework but sharing same hardware setup, virtual environment and cycling interaction. The games in user study 2 and 3 were all built using framework modules.

The evaluation covers two aspects, the framework feasibility and usability evaluation. The
feasibility evaluation is a study that assesses whether the proposed framework can help HCI community in research. Therefore, this evaluation analyses four aspects of feasibility (success rate, exercise performance, motion sickness, perceived competence) through the VR exergames used in user studies. The results show that the hardware setup and interaction design supported by framework modules can contribute to a very low failure (technical issue) in VR exergames for user studies. In addition, the features brought by framework modules can also engage players in doing physical activity, meanwhile motion sickness is significantly reduced. The perceived competence analysis further suggests that the framework modules improve physical activity and game experiences.

The framework usability evaluation studies how the framework satisfies the players as end-users in HCI studies, where a set of factors is analysed. Results have shown that 207 participants provided generally positive feedback on enjoyment, and the immersion feedbacks in user study 3 are overwhelmingly positive. In addition, the exergame interface and controls were rated as intuitive by participants, where they could focus on game play rather than game mechanics. The high ratio of intention for future play proves that the games attracted and satisfied players, which were also indicated in participants’ verbal reactions.

There are multiple reasons that contribute to the results above. On one hand, the framework modules support the latest VR devices, in which the headsets display high resolution and high colour fidelity scenes to reduce motion sickness. On the other hand, our VR Scene Module uses real-world imageries to create the panoramic virtual scene which reduces the dizzy feeling in
3D scenes. Furthermore, our interaction design in *Physiological & Pedalling Device Support Module* also provides a good solution to reduce the motion sickness in exergames, where players sit in trike or a chair to pedal and the cycling is mapped to in-game movement based on the pedalling speed. In addition, our modules provide physiological data as exercise information, which also increases the overall immersion. The games with low motion sickness rate, realistic scenes and exercise state data engage players to cycle to explore the environment. The design and mechanic employed are easy for players to control their own game outcomes and experiences.

From the above analysis we can draw the conclusion that the framework has good feasibility and usability in creating VR exergames for user studies in HCI.

**Limitation**

Firstly, the evaluations only cover four VR exergames that were developed using the framework modules, in which one game (Game 3) did not have a fully immersive virtual scene. Future evaluation should include more exergames built with framework and different game mechanics, which are currently been developed by the students in our laboratory. Secondly, the evaluation on exercise performance should compare the physiological status before and during the exercise for a better understanding on how framework improve physical activity through exergames. However, our data did not include the pre-game physiological data. In our future studies, we need to include as much data as possible for a better measurement result. Thirdly, though we have recruited a large number of participants in previous user studies, we still need
more participants in future studies for achieving more accurate evaluation results. Fourthly, feasibility and usability are not just limited to those aspects listed in the thesis. For example, the TELOS model (Heathcote, 2005) covers five components to evaluate the feasibility. The economic feasibility and legal feasibility are not discussed in this thesis, because the thesis focuses on introducing technical feasibility of this framework as an open-source technique. Future evaluation should include more analysis factors such as schedule feasibility and errors in usability to present a comprehensive understanding of the framework’s feasibility and usability in HCI research.

Conclusion

Based on our previous three user studies, the analysis in this chapter presented valuable insights into how the framework performed in HCI research through the VR exergames. Although there were only four VR exergames developed using framework modules, the user studies have successfully recruited more than 200 participants, which lead to significant findings. The results show that the framework modules facilitate user studies through providing low motion sickness and immersive virtual experience in the VR exergames, as well as intuitive interface for game controls. These advantages improve the enjoyment of exergames that add to the positive research outcomes, indicating the good feasibility and usability of framework modules. However, there are still limitations in our evaluation such as the game samples are few. Therefore, more VR exergames should be developed and further aspects of feasibility and usability should be included in future evaluations.
Chapter 7. Conclusion

Summary

This chapter concludes the main outcomes of the thesis. Based on the research findings, new research questions and directions are proposed for future work.
CHAPTER 7. CONCLUSION

7.1 Outcomes

This thesis has introduced an object-oriented software framework as a tool for HCI community to customise VR exergames. The literature review in Chapter 2 discussed the immersive VR technologies in HCI domain, the generalizations of VR exergame design, and the software framework technique, revealing the great significance of using object-oriented software framework to assist the development of VR exergames. Although researchers and expert programmers in HCI have implemented different frameworks that integrated with diverse hardware and reusable software components for creating VR exergames, elements such as design patterns that abstract domain knowledges, explorable virtual scenes and experiment-oriented features are still missing. Therefore, providing an object-oriented software framework that integrated with design patterns, open-space VR scene with low motion sickness and grouping options in user studies is one of the effective approaches to support customising VR exergames for HCI community.

However, there are two main difficulties in building such a framework: 1) there is barely specific guidance on how to design and implement such a framework in VR exergame domain, where different interaction hardware are commonly involved; and 2) there is no consensus on how to evaluate the quality of framework, its feasibility and performance in HCI studies. In software engineering, through incorporating domain specific knowledges into designs, the component-based software technology has provided solutions to develop the framework’s functionalities such as hardware integration and virtual scene generation. This thesis reports
CHAPTER 7. CONCLUSION

the details of our component-based software framework which uses different modules to provide diverse sub-functionalities of a VR exergame. The evaluation of framework code and performance are also introduced. Hence, this thesis provides solutions to the difficulties through addressing three main questions: 1) it specifies the design and implementation details of a component-based software framework for customising VR exergames; 2) the framework code is analysed from the software engineering perspective to reveal the benefits of effort saving with framework and the quality of its source code; 3) the framework feasibility evaluation and usability evaluation are studied to assess how the framework performs in HCI studies through the exergames built with its modules.

This framework was firstly designed based on our experiences in developing VR exergames for user studies, which targeted to not only provide a tool for our research group in customising future VR exergames but also benefit the HCI community. Through analysing the common architecture of VR systems and domain specific practices, major functionalities of VR exergames such as interaction inputs, VR devices, VR scene and physiological data were abstracted as different software components. These components were then encapsulated into software modules with design patterns. The design specifications of each module (e.g. the manager-behaviour structure) in Unity game engine are introduced in Chapter 3. This chapter also introduces the data flow of the framework in terms of modules, in which modules are bridged through data.

The framework modules were implemented via deriving and extending the core functions
of Unity’s base class. Following the Entity-Component-System design pattern and abstract factory design pattern, the framework module’s code was written into behaviour script (achieve basic functionality) and manager script (bridge and manage different behaviour scripts). All the modules are combined and managed through a framework manager script. Chapter 4 introduces the detail implementations of each module.

As a tool for developing applications, the framework should be used for customising VR exergames in practice and the framework code quality should meet the requirement of standard software metrics. Therefore, six VR exergames were developed and used in our user studies, in which two were built from scratch without using framework and four were developed with framework modules. Chapter 5 introduces these six games and the studies on code analysis. Standard software metrics (e.g. SLOC, IL code) were used to evaluate how the framework helps to reduce efforts in development. Meanwhile standard code metrics such as Maintainability Index, Cyclomatic Complexity, Depth of Inheritance and Class Coupling were used to evaluate the quality of framework’s source code. According to the code study and metrics analysis, we noted that our framework saved effort and time on the integration of hardware and virtual scene into VR exergames. The framework code itself has good maintainability and reusability, indicating a good code structure and reliability in the VR exergames developed with it.

The objective of this framework is providing a tool for generating applications in HCI research, so it is necessary to evaluate how this framework performs in user studies through
the applications customised with it. Consequently, the feasibility evaluation and usability evaluation were conducted in Chapter 6 based on the VR exergames used and data collected in previous user studies. The exergames in all three user studies used the same hardware setup and virtual scenes, while exergames in user study 2 and user study 3 employed framework modules. The feasibility evaluation analysed whether the framework met the requirements of HCI community. Results show that the framework supported high success rate in experiment-oriented applications and provided low motion sickness environment, which improved players’ exercise performance and perceived competence. The usability evaluation assessed how end-users, i.e. players in user studies, were engaged in our VR exergames. The findings revealed that exergames developed using the framework provided intuitive control interfaces, improved player’s enjoyment and contributed to positive experiment feedbacks. The overall results of these two evaluations indicate a good feasibility and usability of the framework modules.

7.2 Future work

The thesis proposes an object-oriented software framework for developing immersive VR exergames with design details and implementation specifications. It also provides evidences for the advantages of using framework modules to create VR exergames, and the positive impacts of using those applications in user studies. The findings are significant, which show the great potential of this framework to be an efficient tool for research. However, there are still some research questions that need to be addressed in future work for a better support for HCI community.
More functionalities in framework modules

Currently this framework has six modules to provide different functionalities of a VR exergame, e.g. VR Device Support Module for adding VR platform into Unity project and Physiological & Pedalling Device Support Module for integrating exercise related hardware into Unity project. Although our modules have covered full body interaction such as head movement, hand interaction and cycling, the diversity of interactions is still limited to the hardware we use. For example, our module only supports the hand interaction through VR controllers (movement tracking and button press response). However, researchers in HCI domain have been using a wide variety of hardware in different research, where interaction tools such as haptic controllers, hand tracking sensors and sensorised gloves are commonly involved (Carrozzino et al., 2005; Hilfert & König, 2016; Zhao et al., 2018). Therefore, it is necessary to support more hardware in our modules to increase the potential of the framework to be used in more studies. In this case, more modules providing different interaction methods should be added in our future work.

Besides the hardware, modules that only involves software functionalities such as VR Scene Module and User Study Module can also be extended. Our VR Scene Module can provide massive Google street view images for rendering the explorable, open-space scene in VR headset, where the scene is panoramic with a 360-degree field of view. Nevertheless, the scene is not three-dimensional, where 3D scene is widely used in VR experiments for multiple interaction possibilities. Hence, it is important to provide an explorable 3D scene in VR Scene Module so that the developer could have alternative choices when creating the virtual scene.
with our framework. Similarly, user study related functionalities in recording more user behaviours such as user goals can also be added into User Study Module for future work.

*More exergames developed with framework*

Our framework has contributed to the development of four VR exergames by the time the thesis is written. Though all the four games have been used in user studies with hundreds of participants recruited, the usage of framework is still limited since the users only involve several students. As a tool that is designed for HCI community, the framework should be employed by as many developers as possible. Therefore, the next step is to let students with cross-disciplinary backgrounds use our framework in building capstone projects. In this way, the capability and performance of our framework can be better revealed through the process from designing to implementation of different VR applications.

*More comprehensive evaluations*

In this thesis, the evaluations consist of framework code evaluation, feasibility evaluation and usability evaluation. Although the code evaluation has revealed that the framework is promising in saving effort and time when building VR exergames with good maintainability, reliability and reusability, the applications created with framework are still too few. Thus, it is essential to build more VR exergames in future, and then conduct the code study with these games for a more accurate performance of framework in development process. In addition, the analysis of framework source code needs to include more standard code metrics for a thorough
understanding of framework quality in future work. Further exergames should also be customised with framework modules for conducting user studies, which collect more player data and include more metrics with different standard questionnaires. Therefore, future studies can disclose a comprehensive understanding of the framework’s feasibility and usability in improving player’s enjoyment and study outcomes through games built with it. In addition, the comparison between the proposed framework and other state-of-the-art VR frameworks could add more edge to the work, which is also another important aspect to explore in future work.

7.3 Final conclusion

In summary, this thesis provides HCI community a software framework for customising VR exergames in Unity. It eases the work of developers in implementation, and facilitates the reliability and maintainability of applications, as well as contributes to positive outcomes in user studies. Furthermore, the use of framework is not limited to HCI domain, it could also benefit researchers in other domains who need VR as a tool for research.
## Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AR</td>
<td>Augmented Reality</td>
</tr>
<tr>
<td>BMI</td>
<td>Body Mass Index</td>
</tr>
<tr>
<td>BMR</td>
<td>Basal Metabolic Rate</td>
</tr>
<tr>
<td>CAVE</td>
<td>Cave Automated Virtual Environment</td>
</tr>
<tr>
<td>CBSE</td>
<td>Component-Based Software Engineering</td>
</tr>
<tr>
<td>COM</td>
<td>Component Object Model</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separated Values</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HCI</td>
<td>Human Computer Interaction</td>
</tr>
<tr>
<td>HMD</td>
<td>Head-Mounted Display</td>
</tr>
<tr>
<td>HUD</td>
<td>Head-Up Display</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MCI</td>
<td>Mild Cognitive Impairment</td>
</tr>
<tr>
<td>MFC</td>
<td>Microsoft Foundation Classes</td>
</tr>
<tr>
<td>MSDN</td>
<td>Microsoft Developer Network</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Controller</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>OLE</td>
<td>Object Linking and Embedding</td>
</tr>
<tr>
<td>OOAF</td>
<td>Object-Oriented Application Framework</td>
</tr>
<tr>
<td>OpenGL</td>
<td>Open Graphics Library</td>
</tr>
<tr>
<td>RMI</td>
<td>Remote Method Invocation</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
<tr>
<td>SLOC</td>
<td>Source Lines of Code</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
<tr>
<td>VE</td>
<td>Virtual Environment</td>
</tr>
<tr>
<td>VR</td>
<td>Virtual Reality</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
Bibliography


Cheng, A., Yang, L., & Andersen, E. (2017). Teaching language and culture with a virtual


Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.


BIBLIOGRAPHY


values?view=vs-2015


demultiplexing and dispatching. In Pattern Languages of Program Design (pp. 529–545).
Addison-Wesley.

communication software. In Handbook of Programming Languages (Vol. 1). Macmillan.


36.

of Virtual Reality 2013 (Vol. 8649, p. 864902). SPIE.

feedback framework for rehabilitation of stroke patients. Informatics in Medicine
Unlocked, 13, 41–50.

Shatnawi, R. (2010). A quantitative investigation of the acceptable risk levels of object-oriented
metrics in open-source systems. IEEE Transactions on Software Engineering, 36(2), 216–
225.

reality exergame design. In Proceedings of the 16th Australasian User Interface
Conference (Vol. 162, pp. 61–68). ACM.

framework for designing training based sports virtual reality performance simulators. In
Proceedings of the 12th Conference of the International Sports Engineering Association
(Vol. 2, p. 214). MDPI.

and Design. Morgan Kaufmann.

Proceedings of the 5th International Conference on Computer Graphics and Interactive
Techniques in Australia and Southeast Asia (pp. 289–295). ACM.

Skjæret, N., Nawaz, A., Ystmark, K., Dahl, Y., Helbostad, J. L., Svanaes, D., & Vereijken, B.


Welker, K. D. (2001). The software maintainability index revisited. In CrossTalk 14 (pp. 18–21). CiteSeerX.


Appendices

Appendix 1: Post-test Questionnaire in User Study 1

Rate the questions on scale of 1 (do not agree) to 5 (strongly agree):

Task enjoyment:
- I had fun playing the VR game.
- I think the VR game was interesting.
- I think the VR game was boring.
- I think the VR game was a waste of my time.

Motivation:
- I would like to spend more time playing this game.
- Given the chance, I would play this game in my free time.

Performance measures:
- I am proud of my results in the VR game.
- I felt excited while playing the VR game.
- I felt stressed while playing the VR game.

Competence:
- The VR game was difficult to play.
- I think I did a good job playing the game.
- It was important for me to do well in this game.
- I tried very hard in this game.

Connectedness:
- Would you play this game with a friend?
- Would you show your tour to friends or family? (In virtual tour condition)

Beta test/improvement questions (short answer):
- What do you think can be improved about the game?
• Would you pay money to play this game (e.g. in a game arcade)? If yes, how much?
• I would recommend this game to a friend.

Appendix 2: Post Questionnaire in User Study 2 including Physical Activity Enjoyment

Scale (PACES)

Please rate how you feel *at the moment* about the physical activity you have been doing:

<table>
<thead>
<tr>
<th>*</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>I enjoy it</td>
<td>I hate it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I feel bored</td>
<td>I feel interested</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I dislike it</td>
<td>I like it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I find it pleasurable</td>
<td>I find it unpleasurable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am very absorbed in this activity</td>
<td>I am not at all absorbed in this activity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s no fun at all</td>
<td>It’s a lot of fun</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I find it energizing</td>
<td>I find it tiring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It makes me depressed</td>
<td>It makes me happy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s very pleasant</td>
<td>It’s very unpleasant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I feel good physically while doing it</td>
<td>I feel bad physically while doing it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s very invigorating</td>
<td>It’s not at all invigorating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am very frustrated by it</td>
<td>I am not at all frustrated by it</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Scale</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>It’s very gratifying</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s not at all gratifying</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s very exhilarating</td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>It’s not at all exhilarating</td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s not at all stimulating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s very stimulating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It gives me a strong sense of accomplishment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It does not give me any sense of accomplishment at all</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s very refreshing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It’s not at all refreshing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt as though I would rather be doing something else</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I felt as though there was nothing else I would rather be doing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Item is reversed scored (i.e., 1=7, 2=6, ... 6=2, 7=1).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Future participation:

- I am likely to try these games in future.
- Would you like to try the games again now? If yes, please free to try it for few more minutes. If no, why?

Motion sickness:

- Did you experience motion sickness during virtual reality session?