

*Investigation of Novice Authoring Interfaces
for Handheld Augmented Reality*

Lawrence James Sambrooks

BInfoTech(Hons)

College of Science and Engineering

8 November 2017

A thesis presented to Flinders University in total fulfillment of the requirements for
the degree of Doctor of Philosophy

Copyright © 2017 Lawrence Sambrooks

No external editor has been used in the production of this thesis. Typeset with \LaTeX .

A picture is worth a thousand words. An interface is worth a thousand pictures.

— Ben Shneiderman

Contents

Abstract	ix
Declaration	x
Acknowledgments	xi
1 Introduction	1
1.1 Research Objective	2
1.2 Contributions	4
1.3 Structure	5
2 Background	7
2.1 Overview of Augmented Reality	7
2.2 Tracking	8
2.3 AR Displays	12
2.3.1 Head-mounted	12
2.3.2 Handheld	14
2.4 Interaction with AR on Handhelds	16
2.5 Presenting AR Content	19
2.6 AR Frameworks	21
2.7 AR Authoring Tools	22
2.8 Summary	28
3 Effects of Handheld Device Form Factor on AR User Experience	30
3.1 Classifying Form Factors	30
3.2 Tasks	32
3.2.1 Task 1	36
3.2.2 Task 2	36
3.2.3 Task 3	37
3.2.4 Task 4	37
3.2.5 Task 5	38
3.2.6 Task 6	38
3.2.7 Task 7	39
3.3 Hardware	40
3.4 Procedure	41
3.5 Hypotheses	42
3.6 Results	42
3.6.1 SUS	42
3.6.2 SEQ	44
3.6.3 Fatigue	47
3.6.4 Subjective Responses	49

3.7	Summary	51
4	Handheld Augmented Reality Authoring Tool for Inexperienced Operators	53
4.1	Architecture	53
4.1.1	Unity	54
4.1.2	Vuforia	55
4.2	Radial Menu	56
4.2.1	Background and Inspiration	57
4.2.2	Final Structure	61
4.2.3	Visual Breadcrumb	65
4.3	AR Mode	66
4.4	Author Mode	68
4.4.1	Transitioning to Author Mode	69
4.4.2	Author Mode Interaction	70
4.4.3	Create Activity	73
4.4.4	Edit Activity	77
4.4.5	Behaviour Activity	86
4.5	Summary	100
5	HARATIO User Experience Evaluation	102
5.1	Tasks	102
5.1.1	Task 1	102
5.1.2	Task 2	103
5.2	Hardware	104
5.3	Procedure	105
5.4	Hypotheses	108
5.5	Results	108
5.5.1	Task Completion Time	110
5.5.2	Task Completion Rate	110
5.5.3	SEQ	118
5.5.4	SUS	119
5.5.5	HARUS	122
5.5.6	Post-experiment Interview	124
5.6	Summary	129
6	HARATIO User Interface Evaluation with Inexperienced Operators	131
6.1	Summary of HARATIO Changes	131
6.1.1	Author Mode Viewpoint Interaction	132
6.1.2	Occlusion Colour Appearance Within Author Mode	133
6.1.3	Behaviour Activity Menu	134
6.1.4	Object Selection	135
6.1.5	Visual Feedback Improvements: Notification System	135
6.1.6	Scene Serialisation	137
6.1.7	Slide Over Menu	137

6.2	Tasks	139
6.2.1	Task 1: Radial Menu	139
6.2.2	Task 2: Behaviour Script Editor	141
6.3	Hardware	142
6.4	Procedure	142
6.5	Hypotheses	145
6.6	Results	145
6.6.1	Task 1: Radial Menu	145
6.6.2	Task 2: Behaviour Script Editor	152
6.6.3	Observations Regarding the Effects of Age	159
6.7	Summary	161
7	Conclusion	163
7.1	Contributions	163
7.1.1	Ideal Form Factor for Mobile AR	163
7.1.2	Simplification of AR Authoring Complexities	164
7.1.3	Interface Considerations for a Novice User Experience	166
7.2	Limitations	168
7.3	Future Work	169
7.4	Final Remarks	170
A	Handheld Device Survey	172
B	Description of Triggers and Actions	175
B.1	Triggers	175
B.2	Actions	176
C	Ethics Approval for Experiment 1	178
D	Ethics Approval for Experiment 2	179
E	Ethics Approval for Experiment 3	180
F	Task 1 Reference Sheet for Experiment 2	181
G	Task 2 Reference Sheet for Experiment 2	182
H	Task Materials for Experiment 3	183

List of Figures

2.1	The Reality-Virtuality continuum	8
2.2	Microsoft HoloLens and Google GLASS headsets	12
3.1	Screen size groupings of AR-capable handheld devices	31
3.2	Illustration of experimental application process	33
3.3	Overview of tasks one to six	34
3.4	Overview of task seven	35
3.5	Handheld devices used: Nexus 10, Nexus 7, and Nexus 4	40
3.6	SUS data for each form factor	43
3.7	Mean SEQ results for each form factor/task combination	45
3.8	Illustration of different device grips observed	46
3.9	Composite render of task five results	47
3.10	Mean subjective rating of fatigue for each combination of form factor/task	48
3.11	Comparison between pre- and post-test rating of hardware factors	49
4.1	Architectural overview of HARATIO	55
4.2	Example of default and customised frame marker designs	56
4.3	Overlapping pie menu designs	57
4.4	Non-overlapping pie menu designs	59
4.5	Initial HARATIO menu concept	59
4.6	Overview of HARATIO's final radial menu design	62
4.7	Exploded view of radial menu components	63
4.8	Linear hierarchical context menu displayed within Microsoft Outlook	66
4.9	Overview of HARATIO's AR mode	67
4.10	Illustration of steps involved in the transition from AR to author mode	71
4.11	Two different approaches to resolving object selection occlusion	72
4.12	Supported touch interactions within author mode	73
4.13	Relationship between background texture, scene content, and viewpoint	74
4.14	Overview of author mode interfaces	75
4.15	Mapping of scene grid cells to physical fiducial marker size	76
4.16	Example occlusion between physical and virtual objects	78
4.17	HARATIO render queue ordering	79
4.18	Microsoft 3D Builder gimbal widgets	81
4.19	Illustration of rotate, scale, and translate menus	83
4.20	Indirect rotation menu designs supporting three axis rotation	84
4.21	Kodu rules captured within Kodu Game Studio	88
4.22	CFG of HARATIO's behaviour language	93
4.23	Overview of behaviour script editor interface	95
4.24	Overview of translation panel and Behaviour activity menu	96
5.1	Occlusion between cardboard archway and tank object	104

5.2	Logitech HD C525 webcam and GoPro camera	105
5.3	Testing materials used for task one	109
5.4	Testing materials used for task two	109
5.5	Task one and two completion times	111
5.6	Screen captures of task one attempts	112
5.7	Completion of task one metrics by form factor	113
5.8	Clarification of hint message in Repeat action configuration panel	114
5.9	Completion of task two metrics by form factor	116
5.10	Screen captures of task two attempts	117
5.11	SUS responses for smartphone, mini tablet, tablet, and PC devices	120
5.12	SUS rating scales	121
5.13	Participant HARUS responses for smartphone, mini tablet, and tablet devices	123
5.14	HARUS factor scores for manipulability and comprehensibility	123
5.15	Hardware factors considered important for mobile AR authoring	129
6.1	Earlier implementation of author mode viewpoint configuration	132
6.2	Original viewpoint control actions	133
6.3	Original depiction of occlusion-coloured objects	134
6.4	Notification messages	136
6.5	Examples of hamburger button use in old and modern interfaces	138
6.6	Overview of slide over menu interface	139
6.7	Task one scene in silhouette, default, and completed states	140
6.8	Task two scene in default and completed states	142
6.9	Testing station setup with materials provided to participants	144
6.10	Zone distribution of the letters <i>A</i> and <i>R</i>	146
6.11	Relationship between task one completion and perceived confidence level	147
6.12	Task one errors	148
6.13	Breakdown of task one completion showing assistance, errors, and failures	149
6.14	Task completion time (in seconds) by experience group	150
6.15	Distribution of task one SEQ responses	152
6.16	Relationship between task two completion and perceived confidence level	154
6.17	Breakdown of task two completion showing assistance, errors, and failures	156
6.18	Task two completion time (in seconds) by experience group	157
6.19	Distribution of task two SEQ responses by experience grouping	158
6.20	Plot of age group against task completion time for task one and two	160

List of Tables

3.1	Handheld form factor classification	32
3.2	Hardware factors believed to be important for mobile AR	50
3.3	Participants' subjective preferred form factor	51
4.1	Summary of available triggers	93
4.2	Summary of available actions	93
6.1	Stepwise behaviour script examples	143
6.2	Erroneous behaviour scripts from task two attempts	155

Abstract

Current consumer handheld devices are ideal platforms for an Augmented Reality (AR) experience. All the hardware necessary is self-contained within a portable form factor, allowing the device to function like a ‘magnifying glass’ to reveal AR content as the user looks *through* the display. Accordingly, decentralised, ephemeral AR experiences will undoubtedly favour handhelds given the ease with which users can retrieve them from a pocket or bag. If mobile AR is to be adopted on a wide scale, users will need the ability to create and contribute content in addition to viewing it. This will require the development of suitable authoring tools that consider novice users who are unlikely to be proficient with technical AR concepts, programming, or 3D modelling.

This thesis reports on the investigation of appropriate novice AR authoring tools for handheld devices, focussing on suitable user experiences. To understand this problem, three distinct questions were investigated: is there an ideal handheld form factor for mobile AR? can the complexities of AR authoring be simplified for mobile device users? and what interface factors should be considered to provide a suitable user experience?

Given the variety of handhelds currently available, a preliminary user study was performed to investigate the effects of form factor on the AR user experience. The study evaluated three distinct handheld form factors comprising a smartphone, mini tablet, and tablet, and found the mini tablet to provide the best overall experience. Tablets were found to be affected by size and weight issues that overshadow any advantages offered by larger screens. The results of this study were used to inform the design of HARATIO, a prototype mobile AR authoring tool developed to investigate user experiences for mobile AR authoring by novice users. This thesis documents HARATIO’s design and development as well as the two independent user studies that were performed to evaluate it.

The first study assessed HARATIO’s overall usability via structured and unstructured tasks. Results revealed promising usability scores and validated the use of a view freezing technique to stabilise the interaction space and manage the physical characteristics of larger devices. Overall, authoring was considered easiest on the mini tablet form factor.

The second study sought novice participants to evaluate HARATIO’s menu and scripting editor interfaces. The menu design was demonstrated to be effective at facilitating access to authoring features involving creating and editing objects. The scripting editor was shown to simplify the definition of object interactivity for non-programmers. High levels of comprehension with the editing interface and scripting language enabled various script implementations to be produced. The study also discovered a difference in the way distinct age generations approached the use of HARATIO: younger participants exhibited exploratory learning styles while older participants were more cautious and cognisant of making mistakes.

Through the development of HARATIO and subsequent user studies, this thesis contributes to a better understanding of user interfaces and user experiences suitable for novice users and mobile AR authoring.

Declaration

I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Lawrence Sambrooks

November 2017

Acknowledgments

I would like to take the opportunity to thank a number of people who have assisted me throughout my candidature. Firstly, I cannot express enough gratitude to my supervisor, Dr Brett Wilkinson. Brett has put up with me throughout my honours and doctoral research studies and has been a constant source of support, knowledge, and guidance.

I'd also like to express my thanks to a number of staff and students within the College of Science and Engineering (formerly the School of Computer Science, Engineering, and Mathematics) at Flinders University—in no particular order: A/Prof Paul Calder, Dr Carl Mooney, Dr Mark Reilly, David Hobbs, Patrick Armstrong, Peter Mitchell, Brad Wesson, and Scott Cabot. Thank you all for your assistance, thoughts, and general conversations. Thanks also to the students, staff, and those outside of the College who volunteered their time to participate in the user evaluations.

To my parents, Alan and Margaret, thank you for your endless support in whatever I do and especially during the course of this journey. Finally, I'd like to acknowledge John Warren who sadly and suddenly passed away during the final year of my candidature. Much like many of the people I have met over the course of my studies, John, a friend and mentor, was always supportive and encouraging of my abilities and contributed to some of my professional habits.

Lawrence Sambrooks

November 2017

Introduction

Augmented Reality (AR) is the process by which computer-generated content is superimposed with a view of the real world and registered in 3D in real time (Azuma, 1997). The technology has its roots in Sutherland's (1968) seminal work in the late 60s on the development of the first head-mounted display (HMD). Unlike the related Virtual Reality (VR), which replaces the view of the real world with an artificial alternative, AR is additive and instead enhances physical environments or objects with information that would otherwise be unperceived by natural senses.

AR content is typically consumed via one of two display types. Evolved from Sutherland's initial efforts, HMDs sit directly in front of the user's eyes and leave both hands free and unimpeded. As these displays are predominantly output-only, their use often involves some form of tethering to auxiliary processing and input devices. This does not imply HMD-based systems have limited mobility; portable, untethered solutions have demonstrated the applicability of wearable AR in mobile environments (Feiner et al., 1997). The alternative to HMDs is handheld devices that have themselves evolved from palmtop computers, Personal Digital Assistants (PDA), and mobile phones. As with HMDs, early use of handhelds for AR also involved tethering to backend systems due to lack of or underpowered on-board hardware. Rekimoto (1995) was among the first to demonstrate a handheld palmtop device being used for AR in what was described as a 'magnifying glass approach' due to the analogy of looking through the device to reveal the augmented content.

Modern handhelds have advanced far beyond the generation of palmtop computers used by Rekimoto to become almost ubiquitous general-purpose computing platforms. Devices now come in a variety of form factors and frequently include multi-core CPUs, one or more cameras, myriad sensors, persistent network connectivity, as well as high-resolution multi-touch displays. Users have the freedom to choose the form factor that best appeals to their own sensibilities, specific use cases, or ergonomic preferences. The capabilities of these devices enable the realisation of self-contained handheld AR without the need for backend systems, achieving the pinnacle configuration in Wagner and Schmalstieg's (2003) categorisation of client/server interaction types. As AR inevitably becomes more mainstream, handheld devices will almost certainly serve as a common, low-cost delivery platform.

It is easy to imagine a future in which many scenarios could benefit from handheld AR. A class field trip to a local museum could incorporate the use of AR on students' tablets to enhance the educational value of visiting various exhibits. Art galleries could implement handheld AR for general visitors, offering an alternative to audio tours that delivers levels of information and interaction beyond what is capable with narration alone. Local city councils could make available historical AR presentations in which residents and visitors alike use their handheld device to easily switch between current and historical perspectives

of their immediate surroundings. Such presentations would assist with maintaining a link to local cultural heritage.

In accordance with the rise in handheld device use and ownership, Schmalstieg et al. (2011) proposed the term Augmented Reality 2.0 (AR 2.0) to describe a future of wide scale mobile AR adoption that incorporates many Web 2.0 attributes (see also Langlotz, Grubert and Grasset (2013) and Langlotz et al. (2014) for discussion on AR browsers that builds on this). One attribute refers to the bilateral flow of information; that is, in addition to being a consumer of AR content, end users should also have the capability to contribute and modify it. Kurkovsky et al. (2012) echoed similar sentiments when discussing current issues with handheld AR, stating it is “*imperative that regular users be able to add their own content on the go and with minimal technical effort.*” If an end user is also a potential author, appropriate authoring tools will be required that de-emphasise technical complexity in favour of providing an accessible, easy-to-use environment for authoring to occur—in other words, *novice authoring tools*. These tools will need to operate on the presumption of limited or no technical experience with AR concepts, programming, and 3D modelling, and go beyond simply extending existing applications or repackaging specialised frameworks. Additionally, such tools should facilitate the creation of interactive content by providing suitable mechanisms for novice users to script behaviours between virtual elements within a scene as well as the physical environment.

With respect to the described scenarios, such tools have the potential to enable regular users without AR expertise to use their own device and directly assume the role of a content author rather than deferring to developers or technical experts. In preparation for a field trip, a class teacher could visit the museum themselves to author exhibit-based exercises, gallery curators could create AR guided tours and easily adapt them as and when exhibitions change, and council representatives could explore the juxtaposition of *then* and *now* media to prototype suitable historical presentations.

The idea of regular users using their own handheld devices to author interactive AR content describes the overall motivation for this research. Key to providing suitable novice-focussed tools are the ongoing investigations into appropriate interfaces and user experiences suitable for handheld AR systems. When this work began in early 2013, studies examining AR usability and simplified authoring interfaces among distinct handheld form factors were limited. This thesis explores these areas through the development of a mobile AR authoring tool and a series of user evaluations. Throughout the course of this work, mobile technology has continued to advance along with an increased availability of devices suitable for self-contained AR use. The interfaces investigated within this thesis remain relevant to such *new* devices and will benefit from evolutionary improvements in mobile hardware.

1.1 Research Objective

This thesis investigates the topic of mobile AR authoring user experiences suitable for novice users. The overall research objective that has guided this work can be summarised as the following:

Investigate appropriate novice interface paradigms and interaction techniques to support AR authoring on off-the-shelf handheld devices. Validate the user experience via a series of experiments.

Appropriate tools to support content authoring are paramount to the success and mainstream adoption of AR; like any medium, it is only as useful as the content available. While professionally-oriented tools exist in the form of frameworks (see section 2.6) and desktop-based authoring applications (see section 2.7), they assume a degree of technical proficiency with 3D modelling and/or programming, limiting their audience, and do not incorporate user experiences simplified for mobile use cases and touch interaction. This is not to say professional tools aren't valuable—they are—however, there should also be a range of solutions available to support regular users who are not technically proficient with these concepts but who nevertheless have an equal interest in exploring and creating interactive AR content.

The rise of handheld devices as 'go-to' mobile computing platforms has been supported by user interfaces and interaction techniques that have evolved to embrace simplicity, moving beyond WIMP-based (Windows, Icons, Menus, and Pointing device) metaphors and mouse and keyboard interaction. Within these platforms, concepts such as file management are often hidden from the user, and the interfaces typically favour providing a single method of accomplishing an action rather than several. Similar principles should also be applicable to AR authoring on handhelds. Rather than adopt a user experience model built on desktop tools, appropriate interface and interaction paradigms should be explored that leverage the strengths of the platform while likewise delivering user experiences suitable for a novice audience.

Building on ideas developed in foundational AR authoring tools like the Designer's Augmented Reality Toolkit (MacIntyre et al., 2004), as well as the notion of a mobile AR 2.0 future in which users assume both consumer and producer roles, this thesis contributes to the body of knowledge on mobile AR authoring by exploring user interfaces and user experiences appropriate for off-the-shelf handheld devices and novice users. The research will present a fresh approach to a mobile AR authoring interface by combining virtual object creation, editing, and visual scripting capabilities under a cohesive and consistent user experience utilising touch interaction. While many types of media are valid as virtual objects within an AR context; such as audio, video, images, text, and 3D models; this work will focus solely on the authoring of simple 3D objects that directly support user evaluations of AR authoring interfaces and interaction techniques. The creation of a comprehensive authoring tool is not required for the purposes of performing these evaluations.

Following from the described objective, three research questions have been examined throughout the course of this thesis:

Question 1 Are there significant differences in the usability of modern handheld form factors for mobile AR? Does a single form factor provide a demonstrable advantage?

Question 2 Can the complexities associated with using a handheld device for AR authoring be simplified to enable novice users without AR, programming, or 3D modelling experience to effectively create content?

Question 3 What interface factors should be considered to provide a suitable user experience for AR authoring by novice users?

1.2 Contributions

This thesis makes contributions to the field of Human Computer Interaction (HCI) and mobile AR through the investigation and evaluation of modern handheld form factors and AR authoring tools designed for novice users. The main contributions are summarised below and have been supported by the following publications in peer-reviewed conferences proceedings:

- Sambrooks, L. and Wilkinson, B. (2015), Handheld Augmented Reality: Does Size Matter?, in 'Proceedings of the 16th Australasian User Interface Conference (AUIC 2015)', Vol. 27, pp. 11–20.
- Sambrooks, L. and Wilkinson, B. (2016), Designing HARATIO: A Novice AR Authoring Tool, in 'Proceedings of the 28th Australian Conference on Computer-Human Interaction', OzCHI '16, ACM, New York, NY, USA, pp. 175–179.

Evaluation of current handheld form factors for mobile AR use

Given the rapid pace of mobile technology advancement, handheld devices have become available in a variety of form factors. These range from devices with three-inch screens, designed to fit comfortably in a pocket, to devices with thirteen-inch screens that have similar physical footprints to a laptop. While such variety provides end users with the freedom to choose devices based on their own needs, it is not clear what effects these differences have on handheld AR use. AR presents a different set of challenges compared with typical mobile use cases as the user must coordinate not only their own interactions with the device but also the device position and orientation with respect to the AR environment. These challenges are further compounded by the ergonomic disparity between different form factors; a smaller device may be practically usable with one hand but a larger device will often require two.

An evaluation of current handheld form factors was performed to determine whether significant usability differences exist and to establish how these affect the design of mobile AR applications. Wagner et al. (2005) previously identified three categories of handheld AR device, but these have been superseded by technological evolution. A survey of current handheld hardware was performed to produce an up-to-date account of form factor categories representative of modern technology.

Devices characterising these categories—smartphone, mini tablet, and tablet—were used to complete a series of interactive AR tasks exploring various types of mobile AR interactions. Results indicate smartphone and mini tablet form factors perform well above the tablet form factor in terms of overall usability. This is mainly attributed to weight, which is still an issue with larger devices. As the existence of its form factor implies, the mini tablet provides the best overall compromise in terms of key hardware attributes—size, weight, and screen size—and was most preferred by participants. Throughout this thesis, the terms *device* and *form factor* are used interchangeably.

Identification of interfaces suitable for a mobile AR authoring user experience

For a novice-focussed mobile AR authoring tool to be successful, an understanding of suitable interfaces and user experiences is required. These aspects should consider co-

herence, simplicity, and the interaction modalities supported by mobile hardware. To investigate this, a prototype AR authoring tool named HARATIO (Handheld Augmented Reality Authoring Tool for Inexperienced Operators) was developed. The tool supports various handheld form factors and is capable of being used to produce interactive AR scenes without emphasis on technical expertise. The implementation of HARATIO's design builds on the work and ideas of existing research into AR authoring, mobile user interface design, and visual programming, as well as observations from the evaluation of end user AR interaction with modern handheld form factors.

The authoring mode is independent of tracker state and provides a stable environment for interactions to occur while also assisting with the management of fatigue onset that often results from the use of larger devices in suboptimal poses. Authoring activities are exposed via an adaptation of a radial menu design that dynamically reconfigures its appearance to conserve screen space and focus user attention on options relevant to the current task.

As the target user for HARATIO is unlikely to possess programming experience, a visual script editor provides a touch-friendly drag-and-drop interface for defining AR scene interactivity. Scripts—known as *behaviours*—adopt a construct that shares similarities to conditional expressions in which the truthful evaluation of one or more *trigger* clauses results in the execution of one or more *action* clauses. The appearance of behaviour scripts is defined by the interplay between assistive labels and graphical symbols that are formatted to appear as a simple sentence phrase. Script readability and comprehensibility is further enhanced by the simultaneous translation of scripts into full natural language sentences.

The user interface components implemented as part of HARATIO were validated via an evaluation of the user experience in which participants completed structured and unstructured tasks using handheld devices representing distinct form factor categories.

Evaluation of interface elements designed for novice AR authoring on mobile devices

Two key user interface components developed as part of this research were evaluated by inexperienced users with no prior programming or HARATIO experience. A small sample of users with limited HARATIO experience also participated for comparison and to assess learnability. The study focussed on the design of HARATIO's radial menu and visual scripting editor and investigated the effectiveness, efficiency, and performance satisfaction of each for completing mobile AR authoring tasks. One task directed users to correct mistakes in an existing AR scene that involved utilising various editing features accessible via the radial menu. Another task directed users to define the interaction of a virtual object by completing a series of scripts using the visual scripting editor. The results provided insights into the way novice users approach mobile AR authoring and the user interface attributes beneficial to assisting them.

1.3 Structure

The remainder of this thesis is structured as follows: Chapter 2 presents a discussion on background work related to this research. The chapter begins with an overview of AR and

the concept of tracking. The chapter then describes prominent AR display types and the role of the user experience in interacting with and presenting AR content. A summary of notable AR development frameworks and graphical authoring tools follows. Given the goals of this thesis, the discussion focuses on handheld systems and the aspects of existing work that have influenced the investigation of novice authoring solutions.

Chapter 3 covers a preliminary user study performed to evaluate the current state of handheld form factors in terms of their usability for mobile AR. The chapter begins with a reclassification of modern devices suitable for mobile AR into distinct form factor categories and then proceeds with a report on the methods used to evaluate them. Tasks were designed to explore typical end-user AR interactions utilising input modalities supported by handheld devices. The chapter concludes with a discussion of the results and their influence on the remainder of this research.

Chapter 4 describes the design and development of a prototype AR authoring tool named HARATIO. The software architecture, user interface, and user experience design are all covered in detail along with an explanation of decisions made throughout the development process to make the tool suitable for novice users. The descriptions refer to HARATIO in its final evolution following iterative improvements resulting from evaluations of the tool. Relevant background work is also covered, particularly in relation to key interface components encompassing the radial menu and behaviour script editor.

Chapter 5 reports on a study designed to evaluate the overall user experience of HARATIO across a range of device form factors. Tasks explored structured and unstructured situations in which participants either recreated an existing scene based on a list of requirements or were afforded the freedom to design and build their own solution to a scenario description. In the latter case, physical objects were required to coexist with virtual objects and be incorporated into the final design. The chapter concludes with a discussion of results and subjective feedback, observations on the use of HARATIO with different form factors, and an identification of usability issues discovered.

Chapter 6 begins with a description of iterative changes made to HARATIO in response to issues identified in the previous evaluation. These changes were incorporated prior to any further evaluations and led to the final evolution. The chapter then reports on a second study that explored aspects of HARATIO's radial menu and behaviour script editor interfaces and assessed their suitability for novice users via tasks exploring the effectiveness, efficiency, and performance satisfaction. In both cases, existing AR scenes containing errors were corrected by participants either using myriad menu operations or the behaviour script editor. The chapter concludes with a discussion of the results and a reflection of the user interface elements implemented.

Finally, Chapter 7 summarises the contributions of this thesis and discusses possible directions for future work. Appendices supporting this research are included following this chapter.

Background

This chapter provides a discussion of existing work in the field of Augmented Reality (AR). Given the goals of this thesis, the discussion covers the role of handheld devices as well as notable development frameworks and authoring tools. Throughout, this chapter will highlight aspects of existing work that relate to this research.

The chapter is structured as follows. Section 2.1 provides an overview of AR and the important concepts of registration and tracking. Section 2.3 describes prominent display types including [head-mounted](#) and [handheld](#). The role of the user experience in terms of interaction and the presentation of AR content is then discussed in section 2.4 and 2.5, respectively. Section 2.6 summarises notable AR authoring frameworks used to develop AR applications. Finally, section 2.7 discusses existing authoring tools that operate as plug-in or stand-alone solutions to deliver abstraction.

2.1 Overview of Augmented Reality

Sutherland’s seminal work on developing the first head-mounted display (HMD) ([Sutherland, 1968](#)) is widely regarded to coincide with the birth of what we now know as Augmented Reality. Although limited to displaying 3D wireframes, Sutherland’s headset incorporated a display technique that allowed the user to view wireframe images simultaneously with the surrounding environment while being worn. Sutherland posited that this would allow content to “*hang disembodied in space or coincide with maps, desktops, walls, or the keys of a typewriter.*”

The origins of the actual term *Augmented Reality* can be traced back to the early 90s. [Caudell and Mizell \(1992\)](#) proposed the expression when describing the development of technology to assist Boeing factory workers in manufacturing and assembling passenger aircraft by superimposing information—wiring diagrams, markings for holes to be drilled or riveted, and so forth—on see-through HMDs. During these early beginnings, AR was considered a variation of Virtual Reality (VR) in which the user was able to view the real world in addition to virtual content.

After noticing the variety and frequency with which AR was being used and defined, [Milgram et al. \(Milgram and Kishino, 1994; Milgram et al., 1994\)](#) sought to establish a consistent method of classification by defining a ‘Reality-Virtuality’ (RV) continuum that classifies environments based on their mix of real and virtual content (Figure 2.1). At the extremums lie real environments, which denote physical reality where no virtual content is present, and virtual environments, which describe completely artificial worlds created by VR systems. Any combination of the two is placed in between as a form ‘Mixed Reality’ (MR). AR consists of a real world environment that is enhanced, or *augmented*, with virtual content. Its position towards the real environment end of the continuum reflects the nature of the view consisting primarily of physical reality. A variation of AR called

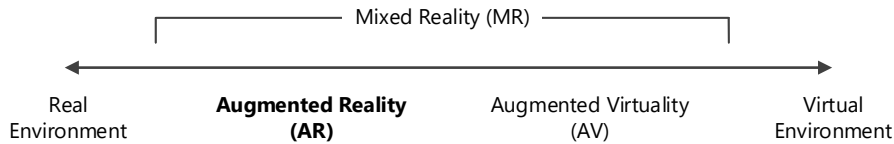


Figure 2.1: The Reality-Virtuality continuum illustrates the relationship between real and virtual environments (Milgram et al., 1994). AR is primarily a real environment augmented with computer-generated content and therefore appears towards the left.

Augmented Virtuality presents an alternate approach by being principally composed of a virtual environment that is embellished with real world content.

In the case of AR, the virtual content needs to be accurately positioned within the real world such that it appears to *belong* there. This must occur in 3D and in real time, which the RV continuum does not represent. A concise definition of characteristics requisite of an AR system was defined by Azuma (1997) and later repeated in Azuma et al. (2001) to comprise the following:

1. Combines real and virtual objects in a real environment;
2. Runs interactively in real time;
3. Registers real and virtual objects with each other in 3D.

Azuma's definition is frequently cited in AR literature and has become a common reference when describing the merits of AR. While modern media, such as motion picture films and television, often incorporate computer generated imagery that is merged with a real world scene, they are pre-rendered and therefore not interactive in real time.

The list of characteristics does not limit an AR system to any one display technology and allows for many different output options. At the time Azuma presented his work, handheld devices mostly referred to Personal Digital Assistants and tethered palmtop displays. Handhelds have since rapidly evolved to become ideal mobile AR platforms that contain all the hardware necessary to deliver self-contained AR experiences.

Key to providing an AR experience that convincingly combines real and virtual content is robust and accurate tracking, which allows virtual objects to be correctly registered within the physical environment.

2.2 Tracking

Tracking allows an AR system to know where it is in the world and provides the reference needed to correctly register virtual content. Registration describes the merging of physical and virtual objects in such a way that the two appear to seamlessly share the same space. When registration is poor, AR systems suffer from visual conflicts that occur when the virtual objects do not correctly align with the physical environment (Azuma, 1997). Conflicts are usually caused by inaccuracies in tracking technology and are easily noticeable. The necessary six degrees of freedom (DOF) tracking required for AR is composed of position (three DOF) and orientation (three DOF) information. Orientation requires higher accuracy than position due to the sensitivity of the human visual system (Azuma, 1993). The effect of registration errors can be seen by the user in terms of view distortion that results in

a misleading or unusable interface (Hallaway et al., 2004; Arth and Schmalstieg, 2011). Because AR systems must operate in real time, maintaining correct registration is not a trivial task. Scene movement resulting from a relocation of the user or physical objects within the environment often requires a re-calculation and re-rendering of the scene, and this must be done quickly enough that successive movements do not produce noticeable lag.

For a mobile AR authoring tool, visual conflicts need to be managed. Creating virtual objects and positioning them with respect to their intended location within a physical environment must be able to occur with a level of registration stability. Holding a handheld device perfectly still while conducting authoring tasks is unrealistic and likely to result in conflicts arising from device movement. The authoring tool developed for the research presented in this thesis (see Chapter 4) implements established solutions (discussed in section 2.4) to address this by enabling the composite AR view to be temporarily frozen in place during the authoring process.

The remainder of this section discusses various approaches to tracking. Vision-based tracking utilises computer vision (CV) algorithms to process sequential image frames in search of features that can be identified. The detection of features usually occurs from fiducial markers or analysis of environmental surroundings. A fiducial marker is an object placed within the real world, usually anchored near or to a point of interest, which acts as a reference to help the system estimate camera pose and accurately determine registration. Though more advanced forms of visual tracking have now superseded fiducial markers, they remain a simple, cost-effective form of tracking aid for localised use cases. The most common format is a black and white (for high contrast) symbol consisting of a distinct inner pattern surrounded by a solid border. Alternate designs are also possible. Fiala (2010) described various fiducial marker designs when discussing robust marker tracking systems.

Even though fiducial markers are simple to incorporate and inexpensive, they do exhibit several disadvantages. Firstly, the target physical environment must be littered with abstract, often unattractive, symbols that serve little purpose other than for the AR system. The need to first prepare the environment is one of the biggest drawbacks of a fiducial setup and limits the overall usefulness of the AR system to the prepared area. Fiducial markers can also be effected by environmental conditions such as precipitation, sunlight (flare), sudden lighting changes, physical obstacles, sharp angles of view, and motion blur (Piekarski et al., 2004). Each of these conditions can reduce tracking accuracy and lead to erroneous detection in the form of false-positives, false-negatives, and inter-marker confusion (Fiala, 2010). A false-positive refers to the case where a marker is detected when no marker exists. A false-negative is the opposite: a marker exists but is not detected. Inter-marker confusion describes the case where a marker is detected correctly but incorrectly identified. These metrics are often used when describing the performance of systems utilising marker tracking.

Feature detection is a more complex form of visual tracking that does not rely on physical fiducial markers. Instead, an image of the environment is searched for naturally occurring features that can be used as descriptors for identification and tracking. Examples include points, lines, edges, and textures. While handheld devices historically lacked the processing power to utilise natural feature detection, Wagner, Reitmayr et al. were able to demonstrate a six DOF feature tracker running self-contained and in real time on mobile

phone and smartphone hardware. Their implementations relied heavily on modifications to SIFT, Ferns, and template matching algorithms, but they were able to achieve interactive frame rates of up to 20 Hz (Wagner, Reitmayr, Mulloni, Drummond and Schmalstieg, 2008) and later 30 Hz (Wagner et al., 2010). Klein and Murray (2009) demonstrated the applicability of keyframe-based SLAM (Simultaneous Localisation and Mapping) on smartphone devices (Apple iPhone 3G) for small environments. Their approach, based on modifications to PTAM (Klein and Murray, 2007), worked around limitations in computational power and camera attributes. Ventura et al. (2014) later explored keyframe-based SLAM and global localisation on smartphones to provide real time tracking in indoor and outdoor environments.

Some degree of marker-less feature detection is now commonplace among popular AR frameworks suitable for mobile devices, such as Vuforia; however, this was not the case when the research presented in this thesis commenced. Implementing technically advanced feature detection was not beneficial to the goals of investigating novice AR authoring interfaces for handheld devices, and so a marker-less implementation was not pursued.

The main alternative to vision-based tracking is sensor-based. These solutions employ sensors in surrounding infrastructure or embedded in devices to provide position and/or orientation (pose) information. They are especially useful in far-field situations where a vision-based approach is impractical due to the distances involved. Perhaps the most common sensor-based tracking solution for positional data in outdoor environments is the Global Positioning System¹ (GPS). GPS receivers decode position information from satellites in three DOF, however, orientation information is lacking making GPS alone unsuitable for AR. Additional sensors; usually in the form of accelerometers, magnetometers, and gyroscopes; can be employed in concert with GPS to provide a six DOF solution. Modern smartphone devices commonly include an array of sensors to support this.

GPS tracking is reliant on signals from overhead satellites and typically fluctuates in accuracy between three and ten meters depending on the number of satellites in use and the connection strength. Accuracy can be improved with differential GPS that utilises additional ground stations with known locations to assist with position calculations (Papagiannakis et al., 2008). The biggest downside to GPS is the fact that it is unusable indoors and in densely covered areas due to satellite signals requiring direct line of sight. In addition to environmental limitations, GPS receivers incorporated into handheld devices are almost always assisted (A-GPS), requiring network access to carrier towers for acceptable accuracy and reasonable satellite acquisition times. This thesis explores a variety of handheld devices ranging from smartphones to tablet devices, the latter of which rarely incorporate GPS receivers. While external units can be added to provide this functionally, they add additional complexity and do not represent hardware setups typically owned by regular users.

Newman et al. (2001) explored a novel indoor solution by implementing a system built with ultrasonic sensors. Users would wear a special HMD with three ultrasonic emitters attached—referred to as ‘bats’—in addition to an inertial tracker. The bats periodically broadcast ultrasonic pulses that were captured by receivers placed at known locations. By measuring the time-of-flight of the bats’ pulse, the position and orientation of the user

¹ GLONASS, the Russian GPS equivalent, is complimentary to GPS and also used.

could be estimated to within three centimetres. As the bats only provided the system with a few updates per second, an inertial tracker was used to fill in the gaps. The system also supported handheld devices. Two bats were used for handheld operation, one attached to the device and one hung around the user's neck. The combination of these bats allowed the system to calculate the direction (pose) the user was looking *through* the device. The bats themselves share similarities to fiducial markers and can even be thought of as a special type of marker; in both cases, the target environment must be pre-prepared. However, the bats are able to offer a distinct advantage over printed fiducial markers in terms of their resistance to environmental effects. While fiducial markers are only useful in well-lit surroundings, ultrasonic bats function equally well in poorly lit settings. Nevertheless, ultrasonic emitters represent an exotic form of tracking aid and require specialised infrastructure and hardware to function, neither of which is widely available in existing environments.

Hybrid solutions have also been explored that incorporate the use of vision- and sensor-based tracking methods. Some of these leverage the strengths of one technique to overcome weaknesses of the other (or vice versa); others overlap techniques to provide more coverage, stability, and accuracy. [Hallaway et al. \(2004\)](#) were among the first to investigate a hybrid solution by combining GPS, ultrasonic, infrared, and dead reckoning to provide coverage between indoor and outdoor locales. The system, a navigational assistant, offered an adaptive interface that dynamically adjusted based on tracking accuracy. [Tinmith \(Piekarski et al., 2004\)](#) similarly combined technologies and used GPS tracking when the user was outdoors and switched to fiducial marker tracking when indoors. In terms of handheld systems, work by [Langlotz, Mooslechner, Zollmann, Degendorfer, Reitmayr and Schmalstieg \(2012\)](#) incorporated Wagner and colleagues' described real time feature tracking in small environments and switched to sensor-fusion (combining visual, accelerometer, magnetometer, and GPS data) in large environments. [Mulloni et al. \(2011\)](#) created a handheld AR navigation system in which strategically placed image markers were used as 'info points' to recalibrate the device's location. In between info points, coarse-grained tracking in the form of dead reckoning was used. *CAViAR* (Context Aware Visual indoor AR) also used dead reckoning to supplement image-based tracking when users moved away from known fiducial markers ([Delail et al., 2012](#)).

While fiducial markers are not the most elegant solution and exhibit several drawbacks, they continue to offer a cost-effective and suitable tracking approach for constrained handheld AR systems in localised settings. By adopting markers for tracking and registration in this thesis, exploration of user interfaces suitable for mobile AR authoring could be conducted with the assurance of reliable and consistent scene generation. Coupled with the relatively low computational demands compared with marker-less tracking, the user experience provided would be less subject to resource constraints involving memory, processing power, and sensor access. Consequently, highly responsive and performant interfaces, key components in their evaluation, are more likely to be attainable. Continued improvements in mobile hardware over the coming years will ultimately absolve the need for fiducial markers and offer similarly efficient, alternative tracking solutions.



Figure 2.2: Microsoft HoloLens (left) provides a stereoscopic view via twin optical combiners. The Google GLASS headset (right) provides a monoscopic view via a small projector mounted above the right eye.

2.3 AR Displays

Various display options exist to view AR content that vary not only in technology but also in terms of whether the display is designed to be ‘egocentric’ or ‘exocentric’ (Milgram et al., 1994). Egocentric displays aim to provide an immersive experience and are normally worn by the user. Conversely, exocentric displays offer windows (or portals) through which content is viewed. Handheld devices typically fill this role, although larger, stationary displays may also be used.

Spatial AR (Raskar et al., 1999) provides an interesting blend between egocentric and exocentric displays as the augmented content is projected directly onto physical objects and therefore does not require the user use an intermediary display to view it. Depending on the level of augmentation, the resulting experience could be considered to resemble an egocentric or exocentric experience.

The remainder of this section will discuss the two prominent display types used for AR output, head-mounted and handheld, and provide a justification for the selection of handhelds as the platform supporting the research presented in this thesis.

2.3.1 Head-mounted

Ever since the pioneering work of Sutherland (1968), head-mounted displays (HMD)—sometimes referred to as head-worn displays (HWD)—have been a popular choice for mixed reality research. They can offer immersive experiences and have the distinct advantage of leaving both hands unimpeded and free to perform interactions or other tasks. Most HMDs are binocular with one or two displays configured to cover both eyes. These headsets completely envelop the user’s field of view (FOV) and provide high levels of immersion. Monocular headsets are also available that utilise a single display that partially or fully covers one eye instead of two. Recent commercial examples of each include Microsoft’s HoloLens and Google’s GLASS (Figure 2.2).

The internal operation of a HMD can use either video see-through or optical see-through technology. Video see-through HMDs present a mediated view in which display output is delivered via one or two cameras mounted directly to the headset. Crucially, if either the display or camera is turned off, the wearer is left blind.

One of the challenges associated with a stereoscopic configuration involving a camera for each eye is ensuring their position on the headset is representative of the wearer’s physical eye location—specifically the inter-pupillary distance (IPD)—so that the view through the headset corresponds to what the wearer would see without it (Edwards et al.,

1993; Drascic and Milgram, 1996). If this is not the case, disorientation can occur resulting in hand-eye coordination issues (Biocca and Rolland, 1998) and perceptual problems that include depth cues being interpreted from the display rather than the scene (Sugihara and Miyasato, 1998; Hoffman et al., 2008).

Although output fidelity is limited by display and camera quality, video see-through headsets do present distinct advantages in terms of the way the final AR scene is composited. Captured images of the real world along with the virtual imagery must all pass through the same rendering subsystem before being displayed. This allows the rendering pipeline to perform additional processing, ranging from correcting end-to-end delays between image capture and virtual object registration to ensuring the final output is at a consistent brightness, contrast, and saturation level. Additionally, occluding virtual objects with physical objects is made easier as complete control of display output is maintained. The same advantages also apply to handheld displays, discussed in the following subsection, and allow for considerable freedom in the presentation of surrounding interface components.

Optical see-through HMDs enable users to see the real world as if looking through a pair of (advanced) eye glasses. They present composite images via the use of optical combiners, which selectively allow some light (the real world) to pass through while reflecting other light (virtual content) back to the user. Compared with a video see-through approach, optical see-through headsets show the real world unmediated; that is, if the headset is turned off or malfunctions, the user is still able to see through it.

A distinct drawback of optical see-through displays concerns occlusion. As some amount of light always passes through, virtual objects never appear truly opaque and instead exhibit a 'ghost-like' appearance (Azuma, 1997). To resolve this, Rolland and Fuchs (2000) suggested using filters or extremely bright displays; in the latter approach, virtual objects would mask physical objects by virtue of being rendered at a much higher contrast. Kiyokawa et al. (2000) explored using transparent LCD panels integrated into the optical combiner setup to selectively block unwanted light. Cakmakci et al. (2004) proposed another approach using a prism and polarisation-based optics to filter light passing through the headset.

Aspects of video and optical see-through techniques have also been applied in novel display setups. Mann's *EyeTap* digital glasses (Mann, 2002; Mann et al., 2005) optically redirects incoming light to a camera via a diverter (a two-sided mirror). The processed light, which represents the scene being viewed, is then synthesised and potentially augmented before being redisplayed via an output device that reflects off the other side of the diverter into the user's eye. Significantly, the light entering and exiting the *EyeTap* is designed to be collinear such that the final view is representative of what would be seen without the device. This overcomes many of the discussed issues with camera placement on video see-through HMDs, including IPD, as the eye sees what it would normally see rather than what the camera sees. Mann has been the epitome case study of his work by wearing evolutions of the *EyeTap* throughout his day-to-day activities. While the glasses have continued to diminish in size, he has noted instances of social resistance to wearing them (Mann, 2012). Similar concerns with social acceptance were raised with the Google Glass headset presented in Figure 2.2 in terms of privacy (Hong, 2013).

Though HMDs afford a level of egocentric immersion unmatched by handheld devices, their use is more intrusive and still burdened by cost, availability, social acceptance, and the

use of additional or unconventional input methods (see section 2.4). Furthermore, from experience, current headsets are not always comfortable or convenient for those who wear prescription eye glasses. Considering the goals of this thesis and the focus on AR authoring tools suitable for novice users, HMDs did not seem appropriate. As discussed in Chapter 1, the motivation for the research presented in this thesis was the notion of regular users using devices they already own, such as smartphones and tablets, to author interactive AR content in many different scenarios. Leveraging existing familiarity with handheld devices as well as their self-contained form factor were both important considerations. Accordingly, HMDs were not explored in detail for this thesis.

2.3.2 Handheld

Handheld displays operate similarly to video see-through HMDs by presenting a mediated view of the physical environment via a camera. They have been described as adopting a ‘magnifying glass’ (Rekimoto, 1995) or ‘magic lenses’ (Bier et al., 1993) metaphor due to the nature in which AR content is revealed as users manipulate the display. Display output is typically presented on ‘monoscopic’ displays although stereoscopic output has also been explored via the use of autostereoscopic displays (Kerber et al., 2013; Berning et al., 2014).

Early handhelds predominately consisted of palmtop computers (Rekimoto and Nagao, 1995) and Personal Digital Assistants (PDA). Processing was often delegated to backend servers while the handheld was used for frontend display and interaction. In classifying the various levels of delegation used by handheld AR, Wagner and Schmalstieg (2003) described four configurations:

1. No delegation; handheld capable of operating by itself (self-contained)
2. Delegation of tracking (assisted tracking)
3. Delegation of tracking and application logic
4. Delegation of everything except camera input and display output (thin client)

The *AR-PDA* (Gausemeier et al., 2003) represented a typical thin client setup (4) in which computationally-intensive tracking and registration tasks were deferred to a backend server via a wireless network connection while the PDA captured user interaction and displayed the final scene. Wagner and Schmalstieg (2003) demonstrated the first system capable of operating self-contained (1) on a PDA; however, the system dynamically switched to assisted tracking (2) when a backend server was available. A self-contained AR implementation running on an unmodified consumer-grade mobile phone was first presented by Mohring et al. (2004). The system could differentiate between different coloured fiducial markers based on interpretation of RGB channel data from the device’s camera stream. A 3D object could then be rendered into the video stream using a weak perspective projection camera model. Later work by Billingham and Henrysson (2006) further investigated the use of mobile phone devices. Initial applications similarly displayed simple 3D models, but follow-up work incorporated interactivity and collaboration. Performance was understandably limited by the hardware and required highly optimised fiducial marker tracking.

Palmtop computers, PDAs, and mobile phones have since been replaced by more capable smartphone and tablet devices that offer multi-touch displays, high-resolution cameras,

myriad sensors, and competent CPUs. The distinction between these devices is comparable to the differences illustrated by Wagner (2007, p.4) in which he defined AR form factors (circa 2007) ranging from heavy backpack HMD systems to increasingly lighter handheld devices comprising tablet PCs, PDAs, and mobile phones. Just as PDAs were positioned between tablet PCs and mobile phones in terms of size and weight, handheld form factors between modern day smartphones and tablets have also emerged. These are colloquially referred to as *phablets*, a portmanteau of the words *phone* and *tablet*. Given the breadth of handheld form factors now available that can serve as functional AR devices, one of the goals of this thesis will be to determine whether one, in particular, is most optimal. This is discussed further in Chapter 3.

While improvements in mobile hardware have allowed for additional computation to be executed on-device and more instances of delegation methods (1) and (2) to be adopted, raw computational performance still lags behind that of PC systems. In an evaluation of CV feature extraction latency for a 1080p video frame between a desktop GPU, desktop CPU, and smartphone CPU, Jain et al. (2015) showed mobile CPU performance trailing by as much as three orders of magnitude. Such a performance deficit limits the use of complex tracking algorithms, especially when the application developer must also be cognisant of resource use so as to preserve battery life and overall system responsiveness. Karaman et al. (2016) profiled the resource use of a sensor-based handheld AR application and demonstrated gains of up to 35% by proposing modifications to the underlying implementations found to consume the most CPU and battery. Resource management remains an important consideration for efficient handheld AR development. As discussed in section 2.2, maintaining a high level of user interface responsiveness was a key consideration for the research presented in this thesis and contributed to the decision to utilise fiducial markers.

AR is rarely at the forefront of the design process when developing handheld devices and, as such, they are not always ergonomically desirable for AR use. Holding a smartphone at arm's length and in an awkward pose can quickly become uncomfortable (Tokusho and Feiner, 2009). In an effort to improve handheld AR ergonomics and comfort, several bespoke hardware designs have been explored in which AR use has been the primary focus.

Mogilev and colleagues' *AR Pad* (Mogilev et al., 2002) was built to explore collaborative AR experiences. The hardware consisted of a six DOF input controller, an LCD screen, and a camera. Users held the device by gripping the controller with both hands and were able to interact with virtual objects using the controller's multi-axis joystick or array of buttons. Tracking and registration were provided by a tethered host computer that processed images sent from the camera and returned the composited AR scene to the screen (delegation method 4).

Veas and Kruijff (2008) developed a custom housing for an ultra-mobile PC (UMPC) designed to improve the ergonomics of handheld AR for spatial interactions. The resulting solution, *Vesp'R*, prioritised human factors and cradled the UMPC between two handles. The handles supported the devices from both sides and assisted in mitigating awkward wrist angles and fatigue. The handles also reduced screen occlusion by embedding input capabilities to limit grip adjustments while performing interactions. An alternate configuration was also supported whereby a single handle was mounted centrally below the device

to facilitate one-handed use. Evolutions of Vesp'R improved robustness and portability while also adding modular capabilities through the attachment/detachment of various sensors (Veas and Kruijff, 2010).

The alternate, single handle configuration supported by Vesp'R shares similarities to the *ARMask* (Grasset et al., 2005). The *ARMask* resembles a lorgnette and was designed to be an integrated solution supporting multimodal interaction via joypad, button, and microphone input. Output was delivered via an integrated HMD, mounted centrally atop the handle, and a headphone system. Just like a handheld device, users could quickly engage with the *ARMask* by moving it into position and just-as-quickly disengage by moving it out of the way.

While the research presented in this thesis targets off-the-shelf handheld devices and does not include exploration of physical attachments to improve device ergonomics, usability factors relating to user comfort for prolonged AR interactions will be considered. Techniques suitable for a mobile AR authoring experience, intended to mitigate potential fatigue, will be supported by analysis of modern handheld form factor usability covered in Chapter 3.

In comparison to HMD systems, one of the key advantages shared by all handheld devices, both off-the-shelf and bespoke, is that users aren't required to wear specialised equipment in order to use them. This affords a seamless and quick transition between viewing the augmented world and viewing the real world by way of the user consciously bringing the device into or out of view. The unencumbered nature of a handheld device allows it to be easily positioned in poses that are difficult or impossible for worn displays. These same traits also encourage and support collaboration. During exhibitions of the *Invisible Train* (Wagner et al., 2004, 2005), a multi-player handheld AR game where users manipulated virtual trains, Wagner and colleagues observed varying degrees of collaboration with handheld AR devices. Some players learnt the game by looking over the shoulder of other players. Others taught the game to peers by passing the device around while providing instruction. Evaluation of another handheld AR game, *medien.welten*, found groups of between two and three high school students effectively collaborated by sharing a single device (Schmalstieg and Wagner, 2007). When each student was provided with their own device, collaboration was still observed with students voluntarily and spontaneously forming groups to share experiences. The ability of a handheld to scale from a singular experience to a multi-user collaborative experience is one of its main benefits over alternate display types.

Handheld AR authoring tools are also likely to foster similar collaborative potential. The example scenarios described in Chapter 1 could all benefit from collaboration in which the process of creating an AR experience is shared among peers to enable real time feedback or assistance. Regular users of smartphones and tablets are already adept at sharing photos and videos with one another, so there is no reason why the same behaviour can't transfer to AR authoring.

2.4 Interaction with AR on Handhelds

Handheld devices are uniquely positioned in that they provide both input and output functionality in the same self-contained package. Consequently, they are more portable,

more open to collaboration, and less cumbersome compared with typical HMD systems. Except for integrated orientation sensors (Mitchell and Wilkinson, 2016) and gaze detection (Park et al., 2008), HMDs typically require auxiliary input devices such as data gloves (Piekarski and Thomas, 2002), tablet PCs (Wilkinson and Calder, 2006), or even haptic guns (Thomas et al., 2002). While the immersion of HMD systems is greater, the coupling of input and output to the same display benefits handheld devices in terms of virtual object selection as it affords an intuitive and direct interaction experience (Wither et al., 2007).

Though direct interaction with handhelds has many benefits, it is also prone to inaccuracy and dynamic registration errors caused by movement of the device or camera jitter from screen interaction. Approaches to address this issue have been explored through techniques that temporarily pause or ‘freeze’ the scene. Guven et al. (2006); Lee et al. (2009); Bai et al. (2012); Vincent et al. (2013) all proposed similar solutions based on temporarily freezing (pausing) camera input and tracking. Freezing the scene allows for a stable *interaction space*—the space where interactions are performed—and additionally permits reorienting the device to a more comfortable pose, something that is beneficial for tasks requiring prolonged interaction or when using larger devices. Once all necessary interactions have occurred, the view can be unfrozen and camera input and tracking resumed. One negative with this approach is that users can become briefly disoriented when the scene is unfrozen if the device pose has changed (Lee et al., 2009).

These techniques are all-or-nothing; that is, the camera feed and all virtual objects within the scene are frozen until the user elects to unfreeze them. Arshad et al. (2016) proposed an alternate solution, called Freeze-Object, that instead implemented freezing on a per-object basis. Using a handheld device, users would locate a virtual object in the scene and then execute an action to freeze only that object, leaving the rest of the content unaffected. Object manipulations could then be performed on the object while the user continued to view the physical environment. This allowed the frozen object to be viewed against different backgrounds without active tracking. A unique implementation for translation involved moving the device to move the object rather than using a menu control or touch gesture—in this case, it was the background that essentially moved, not the object. Freeze-Object appears best suited to smaller devices (smartphones) where the FOV available to the user is limited. By leaving the background active, the authors suggest the wider scene can be scanned to reference the placement of virtual objects against physical objects otherwise out of view. However, larger devices or complex scenes may not be suitable for this approach alone as the real world view of the scene would need to be maintained; placing heavier devices on a table or in a lap to alleviate fatigue and improve working comfort would result in the loss of this reference.

The solutions investigated in this thesis have considered the use of larger devices in authoring scenarios and implemented freezing techniques that support a variety of handheld form factors representative of consumer devices. The proposed freeze method, discussed in Chapter 4, suspends the entire scene view but uniquely allows for manipulation of the virtual component to support object placement (or refinement) in viewpoints not originally captured. By building on existing work and incorporating techniques to stabilise the interaction space during authoring, usability issues regarding device ergonomics (as discussed in section 2.3.2) can be addressed using software solutions that maintain the use of off-the-shelf hardware.

In addition to bringing stability to the interaction space through freezing solutions, alternate interaction techniques, as a substitute or supplement to touch, have also been investigated. Bai et al. (2012) proposed a finger gesture technique involving intangible gestures interpreted using a device's rear camera. This offered an alternate solution to overcoming occlusion and interaction stability by decoupling the interaction from the screen surface. Tracking referenced skin colour to identify the user's hand and then the appropriate fingertips. The approach was susceptible to surrounding lighting changes and so required a well-lit environment for stable detection. Hürst and Van Wezel (2013) similarly implemented finger gesture interaction in an evaluation against touch interaction. Their technique used small coloured markers attached to the fingertips of select fingers to assist with tracking robustness. Touch interaction performed better overall, but finger gesture interaction was suggested to be well suited to games and eLearning applications due to its high 'fun and engagement' factor. Along these lines, Eitsuka and Hirakawa (2013) created a handheld animation authoring tool whereby animations for virtual objects could be described using finger gestures. The finger tracking implementation shared similarities to the work of Hürst and Van Wezel by using a finger cap on users' fingers. This helped reduce the influence of variable lighting on tracking performance, a shortcoming discovered by Bai et al. While these approaches provide interesting and novel alternatives for handheld interaction, their overall usability and usefulness will likely restrict their applicability to specific use cases. For a regular user, these gestures deviate from established input mechanics and would therefore add unnecessary complexity to a novice authoring experience.

Other investigations have explored ways of incorporating physical device movement as a direct alternative or supplement to touch interaction. Gervautz and Schmalstieg (2012) describe this concept as 'embodied interaction'. Billinghurst and Henrysson (2006) summarised a series of mobile phone AR applications using physical device movement for input, which they referred to as the 'Tangible Input Metaphor' (TIM). Initial applications supported viewing a 3D model from various perspectives by moving the device to different viewpoints; the fiducial marker the model was registered against could similarly be moved. Later work focused on active interaction through the applications *AR Blocks* (Henrysson, Billinghurst and Ollila, 2005b) and *AR Tennis* (Henrysson, Billinghurst and Ollila, 2005a). *AR Blocks* allows users to reposition or rotate virtual block objects (akin to Lego). Blocks are selected by moving an on-screen crosshair over the desired object and pressing a button on the phone keypad. Selected blocks can then be positioned relative to the phone, translated by moving the phone, or rotated by rotating the phone. *AR Tennis* is a collaborative game in which two players, each with their own device, hit a virtual tennis ball back and forth using the phones as virtual racquets. Evaluations of the applications resulted in a number of recommendations with regards to designing handheld AR interfaces. These included supporting bimanual input, where users are free to interact with real world objects in addition to the device, and ensuring the chosen interaction methods match device affordances.

Later work by Mossel et al. (2013a) proposed two interaction techniques for handheld AR systems designed for intuitive, single-handed operation that incorporated TIM in addition to touch input. *3DTouch* supports manipulation of virtual objects by combining 2D interactions on the device's touch screen with device pose. For a translate operation,

the device pose is used to determine the translation plane while a one-finger dragging gesture moves the selected object along the plane's x or y axis. Rotation uses the device pose to determine the axis of rotation, and scaling operates similarly to translation with the exception of a second finger being used to determine the scale magnitude. *HOMER-S*, based on *HOMER* (Bowman and Hodges, 1997) but optimised for handhelds, instead uses the six DOF pose of the device to directly control object manipulation. Other than object selection, touch interaction is not required. Translation and rotation occur via a mapping between device pose and the object using a 'virtual hand' metaphor, where the origin point of translation or the pivot point of rotation is determined by the position of the object when selected. Scaling occurs in three DOF instead of six and uses the object's position as a reference point to determine whether scale increases or decreases.

The inclusion of motion sensors in modern handhelds has resulted in increased exposure to their associated input mechanics; however, requiring users learn an interaction language comprising touch and TIM input adds an additional learning penalty to the overall user experience, especially for new users. Furthermore, utilising physical movement of the device conflicts with the goals of providing interaction stability via the use of a freeze technique. Improving the comfort and usability of larger devices by allowing them to be placed on a table or in a lap would be adversely affected when authoring operations required input using device motion. While such novel interaction techniques have informed the direction for the authoring system discussed in this thesis, the style of interaction they enable does not align with the exploration of interactions suitable for novice users and mobile authoring tools that operate on the presumption of limited or no technical expertise.

2.5 Presenting AR Content

The presentation of AR content remains an area of ongoing research. Notable work has explored AR interfaces in terms of the relationship between different device types as well as techniques to display virtual and physical objects together most effectively.

Höllerer et al. (1999) examined the use of hybrid interfaces (Feiner and Shamash, 1991) within a mobile AR system (MARS) by utilising HMD, handheld, and PC devices to present distinct but complementary experiences. Users wearing a HMD could use a handheld device to view a top-down map of their surroundings, which included their current position, and then perform selection tasks that would appear on the HMD. Meanwhile, users using the PC interface could edit objects and collaborate with HMD users to provide them with real time assistance.

The MARS system described the concept of screen- and world-stabilised interface elements. Screen-stabilised elements are positioned relative to screen coordinates and remain fixed in position regardless of viewpoint changes. They typically consist of supporting interface components, such as menus, and are independent of the scene being viewed. World-stabilised elements comprise the virtual content registered with the real world; their rendered perspective changes accordingly with viewpoint changes. Both concepts are applicable equally regardless of display type. For a handheld authoring interface, world-stabilised elements will be most relevant when viewing AR content live where visual aids may be displayed (anchored to the scene) to assist with achieving the desired device

pose in preparation for authoring. The solutions proposed as part of this research will incorporate both concepts.

Similar adaptive AR interfaces have also been explored in novel use cases. The handheld *MagicBook* system (Billingshurst et al., 2001) traversed the RV continuum (section 2.1) from real environment to virtual environment (and vice versa). As the name suggests, the system was built around the act of reading a book. The book contained text and illustrations like any normal book; however, certain pages also contained embedded fiducial markers. If users viewed these pages through an accompanying handheld display—similar in appearance to the AR Mask discussed in section 2.3.2—they could experience the illustrated scenes as pop-out 3D visualisations in AR. If the user wished to further explore a scene, they could choose to fly into it and view it from an immersive VR perspective. What was particularly compelling about these experiences was that the *MagicBook* was multi-user. If one user was immersed in VR, other users viewing the book in AR could view them as a virtual avatar. The reverse was also true: users immersed in VR could look up at those in AR and see them as large virtual heads.

Though this thesis primarily concentrates on a single device type (handhelds), the notion of interfaces adapting to suit different contexts and device characteristics will be considered when exploring suitable authoring solutions. Given the breadth of handheld form factors available, such interfaces will need to effectively scale to accommodate various screen sizes while preserving functionality and usability.

In terms of work related to the presentation of content, Höllerer et al. (2001) proposed three design techniques to assist with the creation of AR interfaces: content filtering, component design, and view management. View management (Bell et al., 2001) relates to the layout and presentation of information with the goal of ensuring virtual object placement does not result in the obstruction of more important objects (virtual or physical). The positioning of text-based annotations is an example. Grasset et al. (2012) developed a view management technique to intelligently position labels with respect to corresponding points of interest (POI). Their approach required no knowledge of the environment beyond the positional data for the label and instead used a visual saliency algorithm together with edge analysis to optimise placement and avoid occluding the associated POI. “*Leader lines*” were used to connect labels to POI.

Content filtering and component design relate to the *amount* and *format* of information. Displaying too many virtual objects in a scene at once may clutter the interface and confuse or overwhelm users with too much information (Azuma et al., 2001; Höllerer et al., 2001; Gervautz and Schmalstieg, 2012; Singh and Singh, 2013). This is especially true for handheld AR implementations where the displays have limited screen space available. Decluttering the interface by removing or filtering non-essential content is one solution that can mitigate this problem. Julier et al. (2000) suggested information filtering as a method to dynamically adjust the amount of content shown at any one time. The determination of what information should be shown was suggested to be based on factors such as physical context, current objectives, personal preferences, and the distance between the device and virtual content. Implementations of information filtering could result in just a label for an object being displayed when far away with full information only being revealed when sufficiently close. Wagner et al. (2005) described a similar idea in the context of a handheld system whereby moving the device closer to a target would allow the user to discover

important details while moving it away would revert to an overview—a magnification metaphor. Looser et al. (2004) implemented manual information filtering within their flat magic lenses whereby users could select between different datasets in real time to adjust the content displayed. A demonstration of a virtual house, viewed through a HMD, enabled the internal wooden structure to be viewed inside the lens and the complete model outside. Their lenses also supported the magnification metaphor. Similarly, *Vidente* (Schall et al., 2009), a handheld AR geographic information system, used a two-step filtering approach to prevent unnecessary screen clutter. Users could select a region of interest from the active AR scene and then toggle relevant information categories on or off to limit the amount of content displayed to a manageable level.

While the concepts of view management and information filtering deal with AR content, the principles they describe are worth considering when contemplating the design of a mobile authoring interface where both content and supporting interface elements need to be displayed simultaneously. Presenting too many interface options at once is likely to clutter the interface (especially on smaller handheld devices), overwhelm novice users, and hinder the view of the AR scene being authored. The solutions explored in this thesis incorporate ideas that share similarities with AR interface techniques such as filtering, element stabilisation, and adaptive interfaces that adjust to different screen sizes.

2.6 AR Frameworks

Frameworks assist with the development of AR applications by providing tracking and object registration capabilities in the form of libraries and Software Development Kits (SDK). They are intended for use by application developers and provide minimal abstraction.

ARToolkit (Kato and Billinghurst, 1999), one of the most popular frameworks, consists of libraries and utilities that enable fiducial markers to be extracted and tracked from a camera stream. Marker data is used to determine the correct position and orientation of virtual objects, and the final scene is rendered back to the video frame as a composite of real and virtual imagery. ARToolkit has continued to evolve since its introduction and is now supported by DAQRI, a commercial AR company. It has remained open source, however, and has been adapted to many different platforms and applications.

Based on ARToolkit, *ARTag* (Fiala, 2005a) improved fiducial marker tracking in areas such as identification reliability (the reduction in the number of false-positives), maximum marker library size, and marker recognition. Whereas ARToolkit used symbols for marker interiors and a template matching technique to determine their ID, ARTag directly encoded the ID within a binary pattern arranged on a six-by-six grid. This approach addressed issues related to marker uniqueness degrading as marker library size increased as well as the associated processing time involved when matching against larger libraries.

Wagner and Schmalstieg (2007) extended ARToolkit with *ARToolkitPlus* to optimise the framework for use on handheld devices. It was used in the first autonomous handheld AR system discussed in section 2.3.2. Like ARTag, ARToolkitPlus supported encoding the marker ID within the marker frame but also maintained compatibility with image-based markers. Fiala (2005b) compared ARTag with ARToolkitPlus over a series of tests to evaluate their respective performance. ARTag was demonstrated to include a number of advantages over ARToolkitPlus, including the ability to recognise partially-occluded

markers, better resist lighting variations, and lower probability of one marker being confused with another (inter-marker confusion). ARTag was subsequently merged into the *Goblin XNA* project (Oda and Feiner, 2012) while ARToolkitPlus was integrated into *Studierstube ES* (Schmalstieg and Wagner, 2007), a variant of the *Studierstube* system (Schmalstieg et al., 2002).

Recently, Vuforia (Qualcomm Connected Experiences Inc., 2014) has emerged as a popular alternative to ARToolkit and its derivatives. As with ARTag and ARToolkitPlus, Vuforia's fiducial marker implementation also adopts binary ID encoding but limits the pattern to the border of the marker, leaving the inner area free for text or graphics. In addition to fiducial markers, Vuforia's tracking capabilities include detection of real world objects and images, although these must first be scanned and uploaded to an online target database before being used. The framework natively supports mobile (iOS and Android), desktop, and wearable (Microsoft HoloLens, shown in Figure 2.2) platforms and is also available as a plugin for Unity (Unity Technologies SF, 2014a). Unity provides a rich library of existing components useful in the creation of AR experiences; these components enable rapid prototyping without the need to implement supporting functionality such as rendering, input, physics, and audio. Vuforia is a proprietary framework and its source code is not available to adapt for specific use cases. Though it must be licensed before use, a no-cost license is available on the provision that an on-screen watermark be visible at all times. It is therefore suitable for academic and non-commercial purposes.

Marneanu et al. (2014) compared ARToolkit with Vuforia, among others, for handheld AR development over a series of tests designed to evaluate their performance in different situations. Examples of the tests included sub-optimal lighting conditions and tolerance to marker size, distance, noise, and angle of incidence. Vuforia was found to perform better than ARToolkit in areas of low visibility, marker noise, and marker distance. The authors concluded the choice of framework would ultimately depend on the circumstances in which it was to be used. As the version of each toolkit tested was not disclosed, the findings may not be representative of recent versions.

At its 2017 World Wide Developers Conference (WWDC), Apple introduced ARKit (Apple Inc., 2017), a framework emanating from their acquisition of Metaio AR in 2015. ARKit supports marker-less tracking of a surrounding physical environment using camera and sensor data (sensor-fusion). The framework is integrated into the 2017 release of their popular mobile operating system, iOS. However, ARKit is only supported on recent Apple handheld devices and, as such, has limited wide-spread application.

Given the goals of this thesis, Vuforia was selected as the AR framework for use throughout this research. Its support for the Unity platform enabled the research focus to remain on exploring authoring interfaces and experiences rather than on implementing underlying application architecture. The combination of Unity and Vuforia additionally facilitated rapid deployment to various mobile platforms, which assisted in conducting evaluations across a range of handheld (and other) devices.

2.7 AR Authoring Tools

While frameworks deliver a great deal of benefit when developing AR applications, they are not end-to-end authoring solutions and require programming knowledge to correctly

implement. Hampshire et al. (2006) proposed a taxonomy of authoring that separated ‘programming frameworks’ from ‘content design tools’ (herein referred to as authoring tools) with the aim of defining different levels of abstraction in the authoring process. Intermediate solutions, such as *APRIL* (Ledermann and Schmalstieg, 2005), have explored abstraction using higher-level descriptive languages (XML) where users can *describe* content rather than *program* it; however, these approaches are still programmatic rather than graphical in nature and subsequently do not incorporate a user interface.

Roberto et al. (2016) performed an analysis of authoring tools emanating from academic and commercial origins to discover trends in authoring strategies. They discovered two paradigms frequently employed: ‘stand-alone’ tools that exist in a self-contained form and ‘plug-in’ tools that extend host software to enable AR features. The remainder of this section discusses existing AR authoring tools that provide abstraction via graphical interfaces. Both plug-in and stand-alone tools are discussed.

Haringer and Regenbrecht (2002) presented the *PowerSpace* authoring system to facilitate the creation of automotive maintenance instructions in AR by acting as a plug-in to Microsoft’s PowerPoint application. Authors could create scenes using standard PowerPoint assets (including text, images, and videos) and the addition of special placeholder elements to reference physical objects in the environment being augmented. The use of PowerPoint provided a high level of abstraction via an easy-to-use interface that was familiar to many computer users and did not require knowledge of AR concepts. Finished slides were imported into the *PowerSpace* Editor where 3D geometry and spatial information were added to produce the final scene. The final scene could be viewed via an HMD or using a companion *PowerSpace* Viewer application.

Like *PowerSpace*, the *Designer’s Augmented Reality Toolkit* (DART) (MacIntyre et al., 2003, 2004) similarly leveraged an existing software application to support AR authoring. The tool allowed designers to directly create AR experiences and rapidly prototype ideas without the need to liaise with developers or learn underlying technical AR concepts. DART functioned as a plug-in to Macromedia Director (now Adobe Director), an interactive content publishing platform, and utilised high-level visual and scripting mechanics rather than low-level programming. Scenes were composed of ‘sprites’ to which scripts, known as ‘behaviours’, could be attached to define properties and interactions. An event system further separated behaviours into ‘cues’ and ‘actions’ that described system events and corresponding operations to perform. The behaviour abstraction provided by DART has influenced the direction of the visual scripting implementation proposed in this thesis. DART scenes could contain video and audio content in addition to 3D models. As with *PowerSpace*, the broad appeal of DART was that users of varying technical proficiency could complete AR projects using existing applications that were familiar (see Gandy and MacIntyre, 2014).

Though these plug-ins lessen the technical expertise required to author AR content, they are bound by the limitations of their host applications, which were never designed for AR authoring. Consequently, they trade the technical complexities associated with AR development for idiosyncrasies of the adapted host software. The solutions investigated in this thesis specifically focus on handheld devices and stand-alone use. Authoring will likewise target a novice userbase but pursue a simplified user experience via a purpose-built interface that leverages existing familiarity with mobile device interaction and established

affordances. The following tools function in a stand-alone capacity and are categorised based on the device on which authoring occurs:

PC

The Component oriented Authoring Tool for Mixed Reality (*CATOMIR*) (Zauner and Haller, 2004) allowed users to visually arrange components and form connections by dragging links between them using predefined *in* and *out* slots. The system included built-in type safety mechanisms whereby only compatible components could be linked—compatible slots were highlighted during the connection process. This idea has been incorporated into the object scripting system developed as part of this thesis to mitigate unintentional misconfigurations. *CATOMIR* was built with the *AMIRE* toolset (Abawi et al., 2004), which was later expanded to include a mobile version, *AMIRE-ES* (Haller et al., 2005), for PDA devices.

The *ComposAR* (Seichter et al., 2008) authoring tool incorporated visual and scripting components within a multi-paned interface. The tool was intended to de-emphasise technical complexity by hiding non-essential aspects of the authoring process. Virtual objects placed in a scene could be manipulated directly or via an interpretive scripting language using Python. Interactions incorporated an event chain system consisting of ‘sensors’, ‘triggers’, and ‘actions’—*sensors* provided data input that could be evaluated by *triggers* to determine whether to invoke an *action*. The tool supported dynamic script evaluation, allowing authors to evaluate scripts and make changes in real time. Although *ComposAR* was initially designed for PCs, it was later modified to support authoring for mobile phones (Wang et al., 2009). This version added another pane to the interface to display a phone keypad as well as the ability to output AR scenes as portable XML files that could be transferred to mobile devices. A companion viewer application was created to allow XML scene files to be loaded and viewed independently on compatible mobile phones.

ComposAR’s use of an interpretive scripting language, while flexible, still places it out of reach for most novice users who lack programming skills. The authoring tool presented in this thesis implements object scripting using a drag-and-drop visual editor that requires no text-based input. However, aspects of *ComposAR* related to the event chain system and support for script evaluation have influenced the direction of the implementation.

SUGAR (System for the development of Unexpensive [*sic*] and Graphical Augmented Reality application) (Gimeno et al., 2013) allowed non-programmers to create marker and marker-less AR scenes for industrial procedures. Uniquely, the tool supported capturing depth information via a sensor such as the Microsoft Kinect to assist with implementing correct occlusion support. *SUGAR* consists of both an editor and viewer component. The editor produced a non-proprietary output file that could be loaded into the viewer on a variety of handheld devices. As with many other tools discussed, *SUGAR*’s editor component was a desktop application; this thesis focuses on novice authoring solutions for handheld devices.

Web

In an educational context, *UNED ARLE* (Cubillo et al., 2015) supported teachers (and students) in preparing augmented content for learning environments. The tool utilised a client-server architecture that comprised a web-based authoring interface and mobile device viewer application. As with many tools seeking high-levels of abstraction, UNED ARLE targets users without programming or AR experience. The web interface enabled various types of media—(animated) 3D objects, text, video, images, and audio—to be added to a central library, which could then be used to build experiences attached to fiducial markers. End-user devices loaded with the mobile viewer application could download content stored in the library and then view it when tracking associated markers. Interaction with content was supported via on-screen buttons or touch gestures. Following evaluation of the tool, the authors suggested teachers' ability to incorporate AR media into educational resources was influenced by their knowledge of the technology. While this thesis explores authoring self-contained on handheld devices, rather than via a separate web interface, similar principles regarding abstraction and simplification have been considered in the investigation of authoring experiences suitable for novice users.

General-purpose AR browser platforms such as *Layar*, *Wikitude*, *ZapWorks*, *Aurasma*, and *AugmentedPro* are also widely available for users of modern smartphone and tablet devices. These platforms provide tools that enable AR experiences to be quickly developed using web- or desktop-based software and then delivered via browser applications installed on end-user devices. While the authoring software is often intended for novice users with little to no programming skills, it is not designed for use on handheld devices directly with touch interaction. In addition to end-to-end solutions, many of these platforms also offer SDKs for more advanced users that wish to integrate their feature-sets into custom developed applications.

HMD

Piekarski (2006) focused on interface and interaction models to directly support AR authoring. Tinmith explored real time egocentric 3D modelling in outdoor environments via a stand-alone HMD-based system. To assist with modelling, especially at a distance, Tinmith implemented the concept of 'AR working planes' (*Piekarski and Thomas, 2004*) that were designed to resemble the working planes used in 3D Computer Aided Design (CAD). Users could utilise AR working planes to create new objects or manipulate existing objects relative to the world, their location, or their head position. Interaction occurred via a data glove and gesture-based system. The palm and each finger of the glove contained a copper strip and forming a connection between the fingers and thumb or fingers and palm invoked a mapped menu command. The thumb of the glove additionally housed a small fiducial marker that was tracked and registered to a 2D cursor capable of being moved along a working plane to support fine-grained interaction. While Tinmith provided a comprehensive platform for in situ authoring, it required users wear specialised equipment that was unsuitable for spontaneous use and arguably intimidating for regular users. Its application was more aligned with specialised use cases rather than the notion of wide scale novice authoring solutions that are the focus of this thesis.

Yang et al. (2016a) explored authoring using a combination of a handheld device and

HMD. In this instance, the system adopted a client-server architecture whereby interactions performed on the handheld were processed by a server before being sent back to the HMD in the form of the user interface. Interactions used the handheld as an input device and consisted of touch gestures and physical manipulation (tilting). In appropriate cases, the handheld was also capable of providing haptic or audio feedback, which could be specified by the author as a form of content interactivity. Though the combination of handheld and HMD represents a novel approach to in situ AR authoring, HMDs have been discounted for this research as discussed in section 2.3.1.

Handheld

Henrysson, Ollila and Billingham (2005) investigated TIM authoring using the physical movement of a mobile phone. Their modelling tool enabled virtual Lego bricks to be manipulated and rearranged in the same manner as their physical counterparts. Manipulation occurred via combinations of device movement to position blocks and physical key presses to select or deselect them. Later work (Henrysson and Billingham, 2007) evaluated six DOF mesh editing using the same ideas. Individual mesh vertices could be translated and rotated by mimicking the desired manipulations with the physical device.

Other modelling solutions have also investigated novel methods of creating and editing virtual objects. *Farrago* (Wozniowski and Warne, 2011) provided a mobile authoring solution in which template objects could be added and positioned using a bimanual process referred to as a '3D object brush'. Users would hold a fiducial marker in their dominant hand, which represented the chosen object, and then create instances of it by tapping the screen of a handheld device held in their non-dominant hand. The initial pose of the object within the scene was determined by the marker pose. The authoring tool presented in this thesis incorporates a similar idea by allowing objects to be rapidly cloned. *Farrago* also supported the creation of short videos based on the real time composition of AR content.

Exploring the idea of video further in an AR context, Langlotz, Zingerle, Grasset, Kaufmann and Reitmayr (2012) presented an in situ mobile video montage tool. Users would record video using a standard video recording device, such as a smartphone, and then upload a georeferenced copy of it to a server where it could be processed to extract the object of interest from the video background. The segmented video clips could then be accessed by other users and registered back into the environment in AR when they moved their mobile device into the vicinity of the video's original geolocation. Langlotz et al. have also investigated the use of audio in AR. Their 'Audio Stickies' work (Langlotz, Regenbrecht, Zollmann and Schmalstieg, 2013) enabled users to accurately place spatially registered audio clips in outdoor environments using a handheld device. Users could view an augmented building depicting an upcoming development and then select a location on it (via the device screen) to create a new audio annotation. Multiple annotations were supported, and colours were used to indicate the state of existing annotations. While the creation of disparate media types is beneficial to AR authoring, the work presented in this thesis focusses on the creation of simple 3D objects that directly support user evaluations of mobile authoring interfaces and interaction techniques (see section 1.1).

The 'AR SketchUp' tool by Langlotz, Mooslechner, Zollmann, Degendorfer, Reitmayr

and Schmalstieg (2012) provided a comprehensive stand-alone authoring solution for inexperienced users on smartphone devices. The tool supported in situ authoring in small (indoor) and large (outdoor) unprepared environments and was built on the concept of AR 2.0 discussed in Chapter 1. As described in section 2.2, the system used a hybrid tracking solution consisting of real time feature detection and sensor-fusion. Authoring tasks were supported by a freeze technique, which has likewise been implemented as a component of this research, to stabilise the authoring environment (section 2.4). Objects could be created as 3D primitives or 2D annotations; the former notably involved a process similar to the SketchUp application whereby the 2D base of the object was first defined and then extruded to form a 3D object. Existing objects could be modified via rotate, scale, and translate operations as well as changes to texture, which could be selected from a built-in colour palette or captured from the host device's camera. *CollARt* (Marzo and Ardaiz, 2013), a mobile AR tool for authoring 3D collages, similarly explored the idea of creating textures using the device camera. Captured images could be manipulated and then used as textures for planar or 3D primitives that were subsequently placed into the scene to form artistic sculptures.

The research motivations of this thesis draw inspiration from ideas proposed in AR 2.0; however, the authoring solutions investigated consider handheld form factors beyond smartphones and focus distinctly on authoring interfaces and experiences. Additionally, means to define per object interactivity are also explored for non-programmers.

Another implementation of authoring in unprepared environments was proposed by Kim et al. (2013) with *IMAF* (Indoor Modelling and Annotation Framework). *IMAF* allowed novice users to use a smartphone to quickly build and annotate 3D models of indoor rooms. After calibrating the tool, users could stand in the centre of a room and model it in five steps: this comprised selecting the floor edges of each wall followed by the ceiling edge of a single wall. Once a room had been modelled, users could add virtual content to it by defining a rectangle, aligned with a floor or wall surface, to act as a reference plane for annotations (text, images, or other 3D assets). The rectangle provided a link between real and virtual spaces and was used to determine the initial pose for annotations. The authoring user experience explored in this thesis also incorporates the use of reference planes to assist novice users unfamiliar with 3D environments in locating virtual objects with respect to other virtual objects or physical objects. Modelled rooms in *IMAF* could be viewed in either an egocentric AR mode or exocentric VR mode.

Rumiński and Walczak's (2013) *MARAT* (Mobile Augmented Reality Authoring Tool) focussed on providing simplicity within a mobile authoring tool while supporting the definition of interactivity without programming. A simplified user interface enabled users to select from a list of virtual objects and assign them to image markers placed next to museum exhibits. Existing objects could be manipulated via a series of touch gestures to rotate or scale them. Basic object interactions could then be added via a series of on-screen wizards that presented the user with a list of available 'events' and associated 'actions'. An event that occurs when touching a virtual object, for example, could be coupled with an action to show a message, play a video/audio clip, or pop-up a quiz. Like *ComposAR*, the concept of separating interactivity into distinct components was incorporated into the scripting system proposed in this thesis. The level of abstraction provided by *MARAT* was considered too inflexible, however, so this thesis has investigated a solution that aims to

provide sufficient simplicity without being too restrictive.

Similar to Farrago, Tang et al. (2015) also explored incorporating marker movement into the authoring process. Their stand-alone mobile AR modelling tool included a default library of 3D primitives that were each represented by a separate fiducial marker. Objects were positioned by moving their associated marker into the desired spatial location. Instances could then be created via selection of a menu option. The interface additionally displayed virtual axis extensions (lines) passing through selected objects in x , y , and z directions. This was suggested to assist users with alignment and the estimation of distances between objects. The authoring interface investigated within this thesis includes similar guiding axis lines to assist with object translation. For novice users unfamiliar with manipulation in a 3D environment, the guidelines provide visual cues in terms of illustrating translation depth and, as per Tang and colleague's suggestion, assisting with object alignment.

Yoo and Lee's (2015) authoring system allowed virtual objects to be created via a series of mnemonic gestures that resembled different object types; a sphere, for example, could be created with a circle gesture. This implementation was suggested to reduce on-screen occlusion caused by menus. Incorporating menu designs that are adaptive to task context and moveable to resolve screen occlusion will also help with this. As with Langlotz and colleagues' tool, a freeze technique was also implemented; however, Yoo and Lee's implementation, called 'multi-freezing', differs by freezing multiple viewpoints. The freeze technique implemented in this thesis explores the manipulation of the frozen viewpoint directly rather than using multiple viewpoints.

More recently, Yang et al. (2016b) investigated using fiducial markers as components in end-user interactions. Their smartphone-based authoring tool proposed 'merge' and 'occlusion interactions' that affected the way virtual content was displayed. The merge interaction allowed new virtual objects to be created when two distinct markers were brought together while the occlusion interaction displayed a virtual object when its corresponding marker was partially occluded. The occlusion feature only supported physical object occlusion by virtual objects; this thesis will also investigate the reverse, which is more challenging as it requires some form of physical environment modelling. Aside from these interactions, the tool also included commonplace rotate, scale, and translate actions to facilitate virtual object manipulation via touch gestures.

2.8 Summary

This chapter has provided a summary of background work in the field of AR, starting with an overview and continuing on to a discussion of different display types, user experiences, authoring frameworks, and graphical authoring tools, the latter of which relates to the goals of this thesis.

As stated in Chapter 1, the objective of this work has been to investigate AR authoring tools suitable for novice users on a range of handheld devices. As these tools should support the creation of interactive AR content, the user interface will need to be designed on the presumption of limited or no experience with AR concepts, programming, and 3D modelling. While existing work addresses aspects of these, there is a gap resulting from a lack of solutions that combine all three. Within this thesis, the existing body of work

identified has influenced the investigation of solutions in terms of usability considerations for mobile interfaces, the types of interactions appropriate, and particular authoring features desirable for the target audience.

Further discussion of existing work related to the progression of the research presented in this thesis is covered throughout remaining chapters. The following chapter reports on an initial evaluation performed to assess the effects of handheld device form factor on the AR user experience.

Effects of Handheld Device Form Factor on AR User Experience

To better understand issues related to mobile AR user experiences, an evaluation was performed to assess whether device form factor had any effect on usability. As discussed in Chapter 1, given the notion that any user with a mobile device is a potential author of AR content, it is expected that a wide variety of devices and form factors could be used. Understanding the respective usability issues and differences between such devices would be useful when considering the design of an authoring tool, especially one intended for novices.

AR typically presents a different set of challenges for handheld devices as the user must coordinate not only their own interactions but also the position and orientation of the device relative to the AR scene being viewed. Much of the work in mobile AR has so far focussed on form factors resembling smartphones, which are an evolution of cellular phones and Personal Digital Assistants (PDAs). This class of device is considered well-suited to ephemeral experiences often attributed to mobile AR due, in part, to its portability and manipulability characteristics. Larger tablet form factors have been previously suggested as being of limited use given their relative size and weight (Arth and Schmalstieg, 2011), but as technology advancements continue to improve on these aspects, their applicability deserves re-evaluation. There are also now in-between form factors—sometimes referred to as *phablets* as a portmanteau of the words *phone* and *tablet*—that purport to offer a middle ground between the two. This device type also warrants investigation.

The work discussed in this chapter was designed to ascertain whether usability differences exist between such form factors and, if so, understand how they might affect user experience with the intent of using the information to guide the design of a mobile AR authoring tool. The chapter begins by classifying handheld form factors, based on available devices, and then discusses the experiment that was performed to evaluate them.

3.1 Classifying Form Factors

The classification of handheld form factors into separate categories is ultimately an acknowledgement of the current state of mobile hardware suitable for AR. In previous years, categories were proposed based on device types considered candidates for stand-alone use. Wagner and colleagues (Wagner et al., 2005; Wagner and Schmalstieg, 2006) identified three categories consisting of cellular phones, PDAs, and Tablet PCs. A later survey of AR research trends by Zhou et al. (2008) described similar categories but also considered newer device types: Tablet PC, Ultra Mobile PC (UMPC), and phone (cellular, smartphone, PDA). These lists describe the devices available at the time. However, with respect to the current state of mobile technology, they are both largely out-of-date. Cellular phones

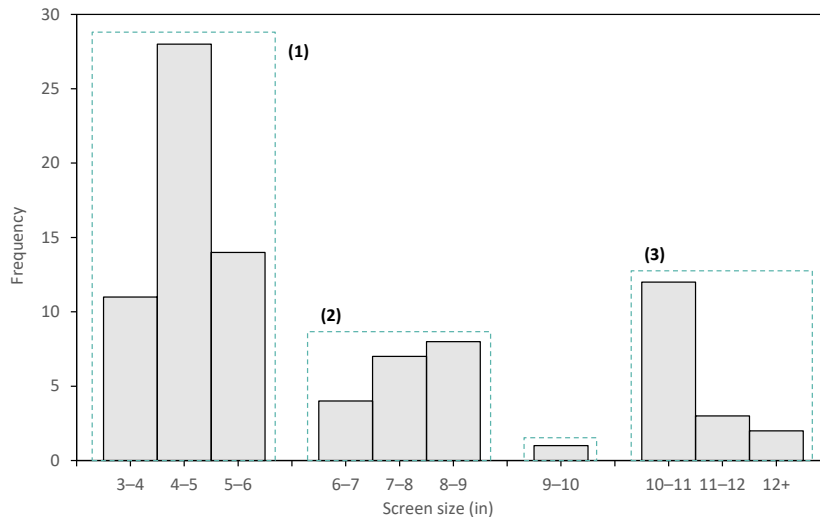


Figure 3.1: Screen size groupings of AR-capable handheld devices (ca. 2014)

and PDAs have been replaced with smartphones that are able to serve the roles of each. Tablet PCs still exist (e. g. Microsoft Surface Pro) but have been supplemented with simpler all-screen touch-based tablets such as Apple’s iPad. The iPad style of tablet is lighter, less complicated to use, more affordable, and built around touch rather than stylus input. UMPCs are perhaps closest to the current range of devices that sit between smartphones and tablets, but even they have been superseded by modern tablets.

Comparing current devices requires an updated categorisation that is representative of modern hardware. It is impractical to test each available device separately, so the use of categories simplifies comparisons by generalising devices into specific classes. To determine the basis for such classes, a survey of hardware commercially available (circa 2014) was conducted with the aim of identifying the sorts of devices users would be likely to own. Between 14 different manufacturers, a total of 90 devices were sampled ranging in screen size from 3 to 12.2 inches. For a device to be considered, it had to be capable of being used for mobile AR. This meant it had to include, at a minimum, a rear-facing camera and an operating system supporting third party applications. As the aim was to establish the range of device classes available, only devices with distinct screen sizes from each manufacturer were considered. Thus, if a manufacturer offered five devices with the same physical screen size, only one was counted. Phone capabilities were not a requirement.

As modern handheld devices are dominated by large touch screens comprising much of their front-facing design, screen size was used as the metric to collate the results. Figure 3.1 shows the distribution of devices by screen size groupings of one inch. While the majority appear to the left of the distribution and represent smartphones, overall there appear to be three main classifications:

1. Screen sizes between 3 and 6 inches (smartphones)
2. Screen sizes between 6 and 9 inches, and
3. Screen sizes of 10 inches or greater

Devices in the first (1) and last (3) categories are well-defined and are commonly referred to in terms of manufacturer marketing as smartphones and tablets, respectively. Devices populating the middle category (2) often comprise those that may be considered either a

Table 3.1: Handheld form factor classification

CLASSIFICATION	ABBREVIATION	SCREEN SIZE
Smartphone	S	3–6 in
Mini tablet	MT	6–9 in
Tablet	T	≥ 10 in

large phone or a mini tablet. While these devices could easily be merged into smartphone and tablet categories, it could be argued that they are distinct enough from an ergonomic and functional perspective to warrant being considered separately. The lone device in the 9 to 9.9-inch group—the Apple iPad—can be consolidated into the tablet category given its 9.7-inch screen size and the fact it is marketed alongside a smaller sibling. For the purposes of the experiment discussed later in this chapter, devices in the middle category are classified as mini tablets on the basis many exhibit characteristics closer to a tablet than a phone. As these devices typically exist as a compromise between a smartphone and tablet, evaluating them separately will help establish whether such compromises suit mobile AR use.

A summary of device classifications, as discussed, is presented in Table 3.1. The boundaries created by the proposed categories enables devices representing each to be used for comparisons. The devices procured for this purpose are discussed in section 3.3. The complete survey is available in Appendix A.

3.2 Tasks

Seven interactive tasks were devised to compare the usability of smartphone, mini tablet, and tablet form factors in various AR scenarios. The tasks were implemented via a custom application developed on the Unity 3D engine (version 4.3; [Unity Technologies SF, 2014a](#)) with the aid of the Vuforia AR library (version 2.8.7; [Qualcomm Connected Experiences Inc., 2014](#)). Although Unity is primarily targeted at game development, the addition of Vuforia enables it to serve as a feature-rich middleware platform on which to build AR applications. In this configuration, the Vuforia library, which comprises a Unity plugin and Software Development Kit, provide the necessary AR tracking capabilities while Unity handles the rendering, physics, interaction, and user interface. Unity and Vuforia are discussed further in Chapter 4.

Though Vuforia supports more advanced tracking techniques, fiducial markers were selected based on their low performance impact, suitability for the testing environment, and intended task placement. Many of the tasks were located on blank walls or surfaces, so there was a distinct lack of features available to support alternative computer vision methods. Vuforia implements fiducial markers in the form of *frame markers*, which encode the ID as a binary pattern within the marker’s border ([Wagner, Langlotz and Schmalstieg, 2008](#)). Apart from task seven which spanned two markers, each task was represented by a single marker. All tasks were displayed on a similar virtual square backdrop that provided a clear view of the task and did not bias the content to portrait or landscape orientation.

As the focus of the experiment was on the differences between device form factors, all tasks were designed around ‘embodied interaction’ ([Gervautz and Schmalstieg, 2012](#)). This refers to interaction via touch screen input or movement of the physical device, the

latter of which is also known as the ‘Tangible Input Metaphor’ (TIM) (Billinghurst and Henrysson, 2006) (refer to section 2.4). In support of this, each task was configured to process interactions independently via its own task loop. Task loops activated whenever their associated frame marker was tracked and continued executing until the task had either been completed or the marker was untracked. The use of independent logic facilitated different combinations of interactions per task and allowed participants to move between them in any order they chose. An illustration of the experimental application process, including the role of the task loop, is given in Figure 3.2.

The remainder of this section discusses each task in more detail. Figure 3.3 illustrates tasks one to six; Figure 3.4 illustrates task seven.

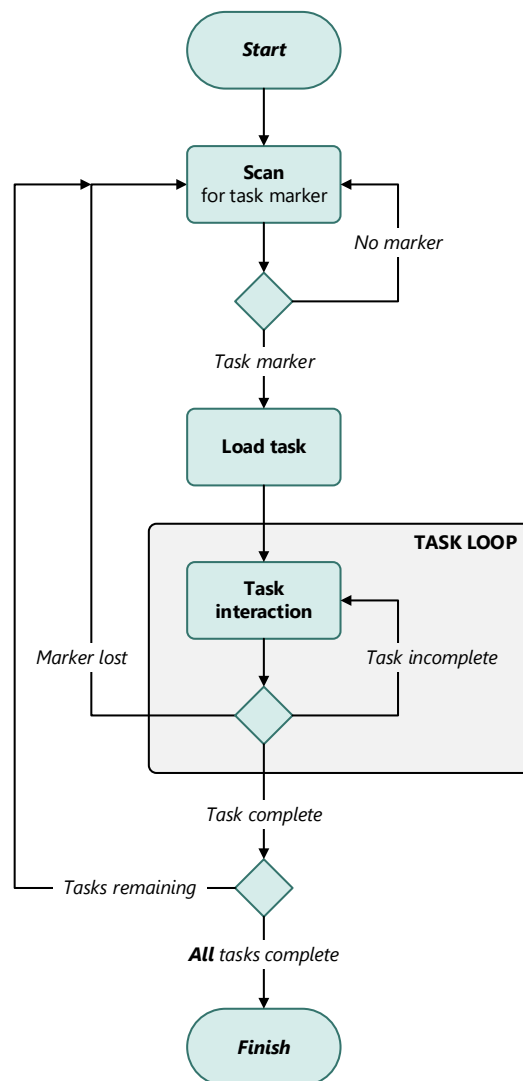


Figure 3.2: Illustration of experimental application process. Each task is assigned to a specific frame marker and processed independently by its own task loop.

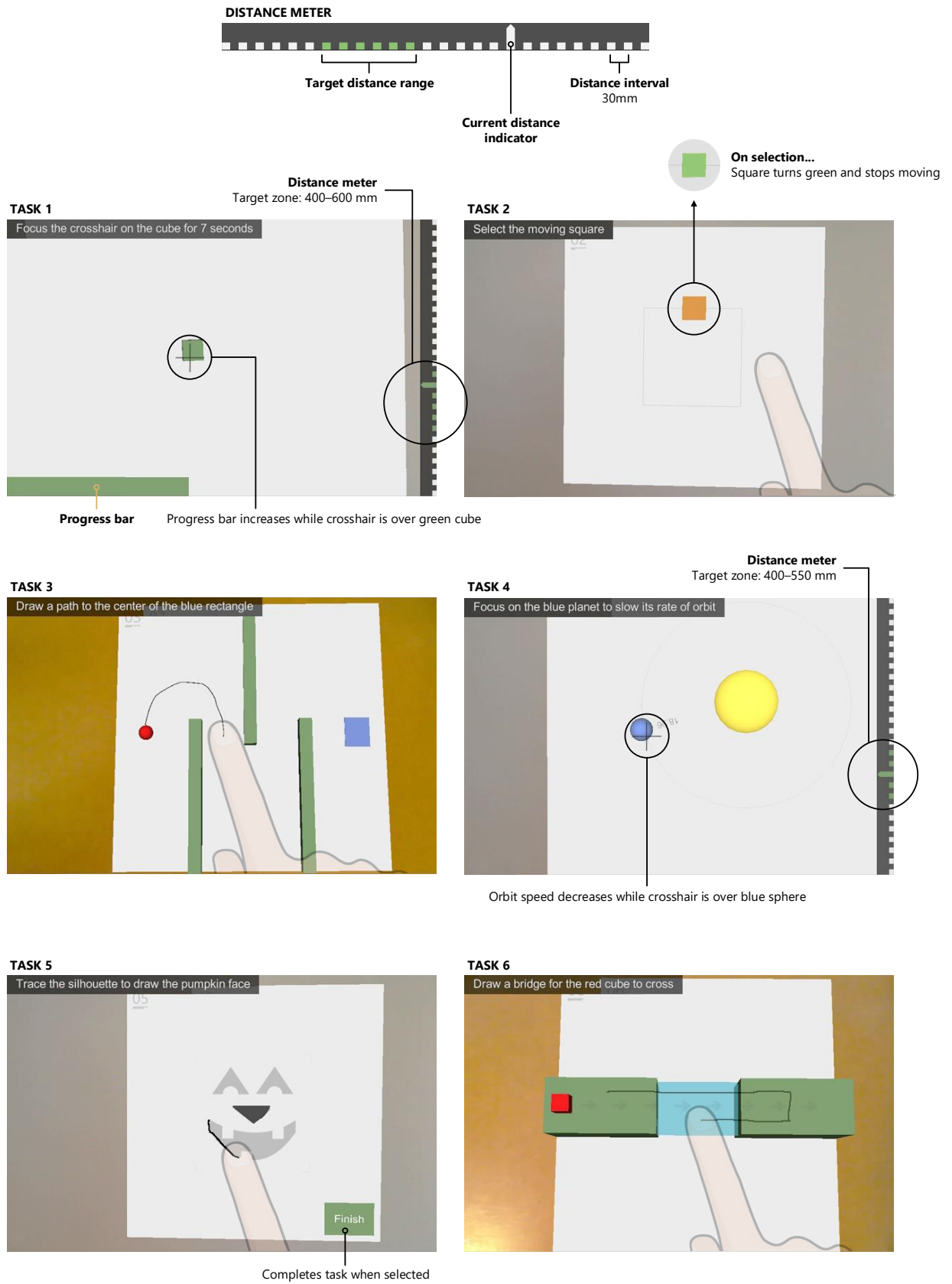


Figure 3.3: Overview of tasks one to six. The distance meter widget, shown in tasks one and four, was used to visually indicate the distance between a device and marker.

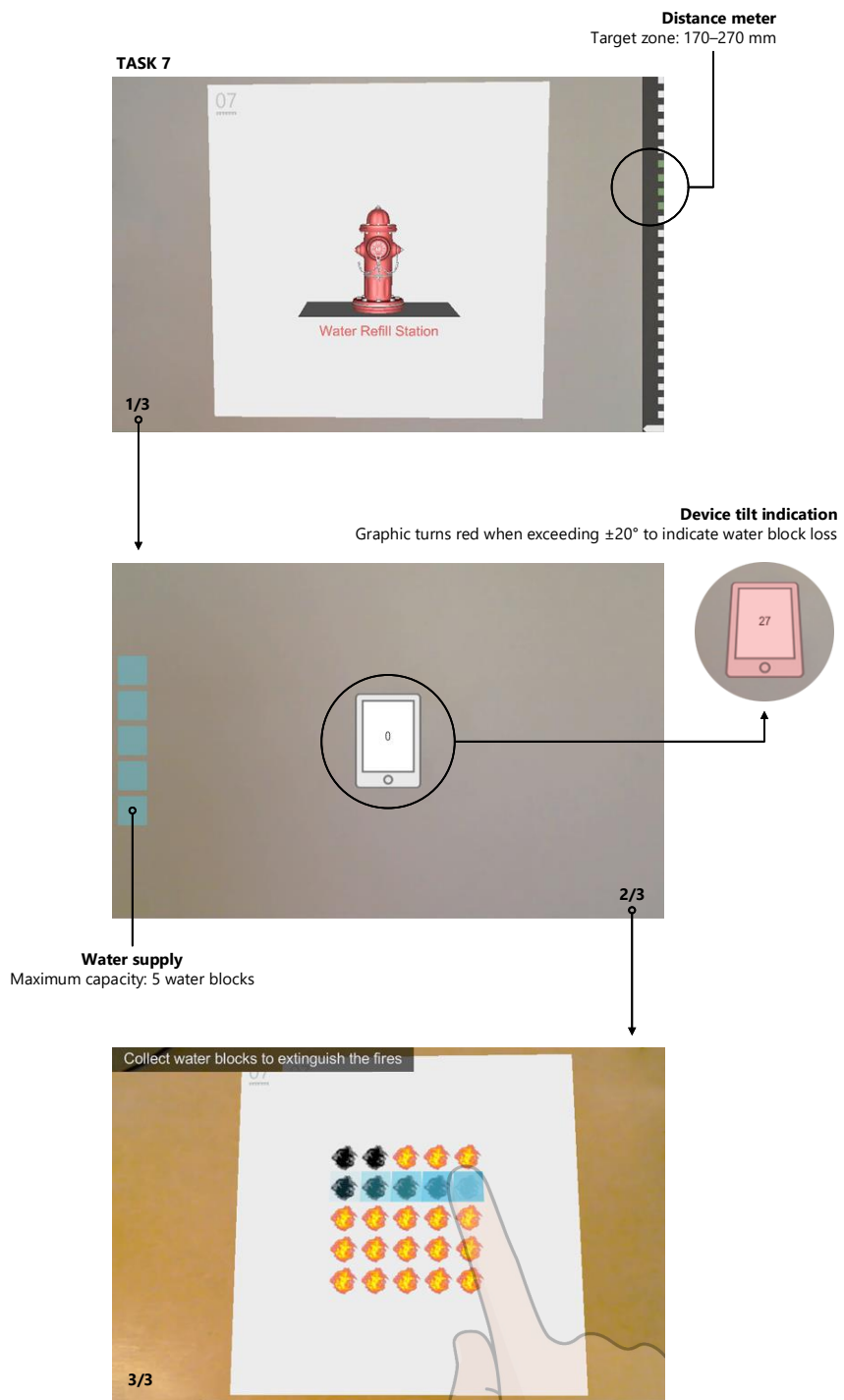


Figure 3.4: Overview of task seven. This task spanned two markers placed approximately 30 m apart. Fire hydrant model obtained from ‘archive3d.net’ (Brook, 2014).

3.2.1 Task 1

The **first** task involved no touch screen interaction and instead asked that participants orient the device so that a crosshair, visible in the centre of the screen, was aligned with a small green 3D cube. Alignment of the crosshair and cube tested participants' ability to maintain device stability. The task was considered complete when the crosshair and cube remained aligned without interruption for 10 seconds. If misalignment did occur, the task would reset and the participant could try again. Informing the participant of task progress and the alignment time remaining were achieved via the combination of a countdown notification displayed in the top left of the screen and a progress bar displayed along the bottom. The appearance of both was determined by the current alignment state. The use of a progress bar was intended to allow for peripheral status monitoring without detracting focus from the crosshair/cube.

To make comparisons between devices more meaningful, and to make the task non-trivial, the device had to be positioned within 400 and 600 millimetres of the marker, which was located on a wall 80 centimetres above the floor (approximately waist level). The distance was calculated based on two positional vectors, one representing the marker and the other the virtual scene camera (that is, the physical device). Adopting a unit scale for Vuforia frame markers within Unity that corresponds to physical marker size¹ yields a usable distance estimation based on the difference between the magnitudes of these vectors. The current distance along with the target distance range were both displayed visually down the right-side of the screen via a custom distance meter widget (Figure 3.3). The widget comprised a series of coloured blocks that represented distance intervals of 30 millimetres. A single elongated block denoted the current distance of the device from the marker. Green coloured blocks indicated the target distance range. The elongated block moved up or down the scale depending on device movement: up as the device moved closer; down as it moved farther away. The distance requirement became satisfied once the current distance was within the target distance range. Participants needed to be cognisant of device distance when attempting the task so as not to move outside the target range. As with misalignment, moving outside the target range triggered a task reset. The use of a visual scale provided peripheral monitoring of distance similar to the aforementioned progress bar. It is simpler for users to gauge distance in this format rather than compare a readout of current distance against target distance values, which would need to be remembered or consulted.

3.2.2 Task 2

The **second** task involved positioning the device above eye level and interacting with the touch screen. The task marker was located on a wall 177 centimetres above the floor. The task was intended to test participants' ability to grip a device in an undesirable pose while simultaneously performing an interaction. Due to the nature of AR content being viable in any physical location, this task mimics the case where users may wish to engage with content that is situated outside of their normal field-of-view. Example situations include in situ authoring, where the position of target objects may not be under the direct control of the author, or users whose physical stature requires they view content looking up or

¹ Frame markers used for the experiment were 14 cm²

down at it. As the task would require the device be secured with one hand, it would also evaluate stability from a different perspective.

The task comprised an orange square (2D plane) that continually moved, clockwise, along a square path. To complete the task, participants had to select the square by tapping on it. Once the square had been selected, it changed colour to green and stopped moving. The movement speed of the square started out reasonably fast, completing a lap of the path approximately once every five seconds. If the user failed to select the square on two consecutive attempts, the speed was subtly decreased such that a lap of the path took an additional two seconds to complete.

3.2.3 Task 3

The **third** task explored the effects of localised interaction occlusion. The task consisted of a simple maze comprising three green walls (3D blocks). A red 3D sphere was placed at one end and a blue exit area (a 2D plane) at the other. Participants were asked to navigate the red sphere through the maze to the exit area by drawing an appropriate path with their finger. The drawing interaction was required to start from the sphere and was illustrated on screen as a black 3D segmented line. As soon as the path interaction finished, signalled by the participant's finger lifting off the screen, the sphere would begin following the path. During this time, all screen interaction was disabled. If the sphere encountered a green wall while it travelled along the path, the task would reset—the sphere would return to its starting location, the participant's path would be cleared, and interaction would be enabled. Completion of the task required the sphere to be navigated to the blue exit area without colliding with any green walls. The task marker was placed on a table at a height of 75 centimetres.

The trade-off between device form factors was of particular interest. As smaller devices are lighter, they are likely to be easier to grip with one hand, but they are also more susceptible to interaction occlusion and the small screens may make drawing the desired path more difficult. Conversely, while larger devices provide more screen real estate to view and interact with a scene, potentially making the act of drawing a path easier, their relative heft may make them more difficult to hold securely and result in perceptions of fatigue.

3.2.4 Task 4

The **fourth** task shared some similarities with the first and second. It was intended to explore form factor manoeuvrability in terms of interaction via controlled device movements. The task consisted of two spheres, one representing a *sun* and another a *planet*. The sun sphere was coloured yellow and appeared larger than the planet sphere, which was coloured blue. The blue sphere continually orbited the yellow sphere in a clockwise direction and completed a single revolution approximately every 18 seconds.

Participants were asked to align a crosshair, displayed in the centre of the screen as per task one, with the orbiting blue sphere by positioning and orienting the device as necessary. The alignment would need to occur between 400 and 550 millimetres of the marker, which was located on a wall 147 centimetres above the floor, so the distance meter described in section 3.2.1 was made available on the right side of the screen. Achieving correct

alignment would result in the orbit speed slowly decaying until it stopped, at which point the task would be complete. Successful completion would involve continual adjustment of device position and orientation, not only to track the blue sphere as it orbited the yellow sphere but also to account for orbital speed changes. A value representing the current orbit speed appeared anchored to the blue sphere as it moved. This value would decrease to show the alignment state as long as the crosshair remained aligned with the blue sphere. If misalignment occurred, the orbit speed would quickly increase back to its original value, replicating a task reset condition. Quick realignment would resume the orbit speed decay at its current value, so participants were not required to start over if they momentarily lost concentration or alignment. A complete reset, in this case, was considered too harsh.

Like task two, subtle assistance was provided if more than two misalignments occurred. Instead of adjusting orbit speed, however, the diameter of the blue sphere was increased by 10%. This could occur up to five times for every two misalignments. As the operation was compounding, the blue sphere could be enlarged to a maximum of 1.6 times its original size.

3.2.5 Task 5

The *fifth* task presented participants with a free-form drawing exercise in which they were asked to trace the silhouette of a pumpkin face by creating the necessary four shape components. The task marker was located on a wall 163 centimetres above the floor and was designed to evaluate prolonged interaction accuracy with the touch screen in suboptimal orientations. While the trade-offs between devices would be comparable to task three, the effects would likely be more pronounced.

Drawing each face component was accomplished by first tracing a path representing the desired shape, similar to the interaction required in task three. When the participant lifted their finger off the screen, the path would be closed (the first and last point joined together) and the data used to construct a 2D plane object representing the path. The plane would then be positioned in the exact location where the path was drawn. Paths were not allowed to intersect and would be ignored if they did. Once a path had been created, it could not be deleted; however, participants were permitted to reposition the resulting plane to better align it with the silhouette via a drag interaction.

Task five has no defined complete state (nor reset state). Participants decided when to complete the task by tapping a *Finish* button that was displayed in the lower right corner of the task area.

3.2.6 Task 6

Task *six* presented a similar drawing exercise to task five but with a more desirable orientation (horizontal as opposed to vertical). The task marker was placed on a table at a height of 75 centimetres. Rather than repeat the same drawing exercise, participants were asked to create a bridge between two green 3D blocks that a red 3D cube could then use to cross between them. Participants defined the bridge in the same manner as they did the pumpkin face shapes. As a silhouette was not provided, the bridge could be any shape desired so long as it remained within the boundary extending over and between the two blocks.

Once the shape of the bridge had been defined and the corresponding 2D plane object created, the red cube would begin moving along a pre-defined straight path (left to right) between the two green blocks. As with task three, touch screen interaction was disabled while movement occurred, and so the bridge shape could not be modified. If the bridge was sufficient, the red cube would successfully cross over to the other green cube and the task would be complete. If the bridge was insufficient, the red cube would likely fall into the blue area beneath—symbolising a body of water—causing the task to reset. This involved the red cube returning to its starting location, the bridge shape being deleted, and interaction re-enabled. The participant could then try again.

3.2.7 Task 7

Task seven deviated from the format of the previous six tasks by being distributed between two physical locations, each with its own marker. The task objective revolved around a mini game in which participants were asked to extinguish a series of 25 small fires. The fires were displayed as graphical sprites within a five-by-five grid on one of the markers placed on a lectern at a height of 100 centimetres. In order to extinguish the fires, participants had to visit the second marker, located approximately 30 metres away on a wall 147 centimetres above the floor, to collect virtual water blocks. Collecting water blocks involved being within 170 and 270 millimetres of the second marker and remaining there while the supply of water blocks was replenished. The current supply was displayed as a series of blue squares vertically down the left side of the screen and replenished at a rate of one every two seconds. A maximum of five could be carried at any one time, and this was indicated as a percentage in the top left corner of the screen. Once the desired number of water blocks had been collected, the participant would make their way back to the first marker (with the fires) and proceed to use the collected water blocks to extinguish them. Extinguishing fires with water blocks could be achieved by tapping on or dragging across them. Fire sprites changed appearance once extinguished to inform the participant of their progress. One water block was consumed for every fire, so the described process of moving between markers, collecting water blocks, and extinguishing fires would have to be repeated at least five times. The task was completed once all fires had been extinguished. While the task may have seemed repetitive, the intention was to explore the perceived portability characteristics of different form factors, how well they worked in motion, and whether participants felt any discomfort or fatigue from using them over a protracted period of time.

During initial pilot testing of the task, it became apparent that some users, after collecting water blocks, would simply walk back to the fires marker with the device held at their side. It was originally anticipated that participants would embrace a metaphor of carrying a bowl of water while moving between markers, so this behaviour invalidated much of the interaction-in-motion of the device and resulted in the distance between markers being artificial rather than meaningful to the task's completion. In response, the task was modified to incorporate a TIM interaction whereby, after collecting water blocks, the device had to remain within $\pm 20^\circ$ perpendicular to the ground to avoid water being 'spilled'. This essentially enforced the metaphor by replicating the state of a physical bowl of water being carried. Tilting the device beyond this threshold resulted in one water



Figure 3.5: Handheld devices used: From left to right: Nexus 10, Nexus 7, and Nexus 4

block being lost every two seconds—the same duration used to replenish water blocks—simulating the act of water spilling out of the bowl. The current tilt of the device was displayed within a graphic whenever at least one water block had been collected and the water marker became untracked. The graphic skewed in accordance with physical device tilt and changed colour to red whenever the threshold was exceeded. This is illustrated in Figure 3.4, which shows the graphic in both states. The graphic disappeared as soon as either the fire or water marker were tracked. A grace period of two seconds, in accordance with the loss rate, allowed minor deviations to be quickly corrected without penalty. Water blocks that were lost would need to be replaced by revisiting the water marker. Given this modification, participants would be required to monitor the device pose as they moved between the water and fire marker to maintain a valid tilt angle. Depending on the number of water blocks lost, this could result in more than five trips being made. Consequently, there was no requirement that five (the maximum) water blocks be collected every time the water marker was visited; the participant could elect to collect one if only one was lost.

3.3 Hardware

Based on the classification of handheld form factors discussed in section 3.1, three devices were acquired representing smartphone, mini tablet, and tablet categories. At the time of testing, Google’s Nexus line of products comprised devices befitting each category: a 4.7-inch Nexus 4 smartphone, a 7-inch Nexus 7 mini tablet, and a 10-inch Nexus 10 tablet. These devices were representative of typical hardware for the time, and all shared a common operating system and similar feature set. The commonality between devices was beneficial to ensuring a consistent AR software experience could be provided.

A proportionally correct size comparison between the devices is given in Figure 3.5.

3.4 Procedure

A total of 15 participants were recruited from the School of Computer Science, Engineering, and Mathematics at Flinders University² to take part in the experiment. Of the 15 volunteers, 14 were male and 1 was female. The average age was between 21 and 30.

The experiment used a within-subjects design with participants completing identical sets of tasks using each of the three handheld devices. Prior to attempting any tasks, participants were provided with a copy of the Tasks Overview Document (TOD) and instructed to complete a pre-test questionnaire. The TOD was produced alongside the tasks to serve as a reference and introduction. The document described a method for completing each task in a predominately pictorial manner and was intended to be perused before any tasks were attempted to mitigate learning effects. The document also outlined important user interface widgets that would be required for certain tasks, such as the distance meter and crosshair. As the goal of the experiment was to compare the usability of different form factors rather than the usability of the AR application, an effort was made to ensure participants would be familiar with the tasks before they attempted them. Revealing the tasks via the TOD meant the focus could remain on the use of each device rather than on learning the application's interface or the specifics of each task. The pre-test questionnaire collected demographic information as well as current device ownership and previous experience with AR. The last three questions comprised a subjective rating of hardware factors that were believed to be important in a mobile AR device, a ranking of how well the participant thought each form factor would perform, and an indication of which they considered preferable overall. These questions were answered with full knowledge of the tasks and devices; they would be revisited after the experiment in the post-test questionnaire.

Following the pre-test questionnaire, participants were shown a map of the testing environment illustrating task (marker) locations. Apart from the final task, which spanned two markers, each task involved visiting a single marker. Participants were handed a single device and instructed to move to the start marker to begin. The start marker ensured all participants began from the same location. The AR scene associated with the marker presented a simple `Start` button that enabled all tasks and initiated background logging. The act of beginning the experiment in this way also served to demonstrate how tasks could be approached (tracked) and viewed with the device.

During the experiment, we wanted to capture participants' reactions towards the device form factors as they were being used, and so actively encouraged the think-aloud protocol (Lewis, 1982). Comments were digitally audio-recorded by the test moderator, and this was supplemented by hand-written notes on observations and nuances that audio could not capture. Video recording was dismissed due to the impracticalities associated with maintaining a usable camera angle on each participant's interactions as they moved between tasks.

Following each task, participants were asked for a verbal response to the Single Ease Question (SEQ) (Sauro, 2012, p.214). The SEQ rates task performance satisfaction and asks users how easy or difficult a task was to complete using a single Likert scale question.

² Any experiment involving human participants from Flinders University requires appropriate ethics approval from the university's Social and Behavioural Research Ethics Committee (SBREC). Appendix C contains a copy of the final ethics approval notice.

Sauro and Dumas (2009) found it to yield reliable results and perform just as well as SMEQ (Subjective Mental Effort Questionnaire) and UME (Usability Magnitude Estimation) questionnaires provided the sample population exceeds ten. Participants provided their response on a scale of one to five with one representing ‘very difficult’ and five ‘very easy’. Once all tasks had been attempted, participants were asked to complete a System Usability Scale (SUS) (Brooke et al., 1996) questionnaire to rate the device form factor’s overall usability. The SUS questions were unmodified save for the replacement of the word *system* with the word *device* and clarification of *various functions* mentioned in question five to include features relevant to handheld hardware such as camera, screen, and audio.

The process was then repeated two more times with the remaining devices. The order in which the devices were used was counterbalanced between participants to avoid ordering effects. After using all three handhelds, participants were asked to complete a post-test questionnaire. The post-test questionnaire sought ratings of form factor enjoyment, difficulty, fatigue, and repeated the three questions discussed earlier regarding form factor features, performance, and subjective preference.

3.5 Hypotheses

The experiment tested three hypotheses that each focussed on aspects of handheld form factor use relevant to mobile AR usability:

- H1. Considering its prevalence as a common form factor throughout mobile AR literature, the smartphone achieves a higher SUS score than both mini tablet and tablet form factors.
- H2. Participants subjectively prefer to use the smartphone over the mini tablet and tablet.
- H3. Tablet weight negatively affects task performance satisfaction (SEQ) in situations where the device must be supported by the user’s hands and arms only.

3.6 Results

This section presents empirical results and analysis of participant data obtained from completion of the seven tasks. Responses to post-task/device questionnaires are covered first followed by discussion of subjective factors including device fatigue, form factor attributes, and pre- and post-experiment preferences.

3.6.1 SUS

A SUS questionnaire was administered following the use of each device. Results from the questionnaires were put through the following formula to determine an overall usability score between 0 and 100:

$$\left(\sum_{i=1,3\dots n}^{n=10} (C_i - 1) + \sum_{i=2,4\dots n}^{n=10} (5 - C_i) \right) \times 2.5 \quad (3.1)$$

As explained by Brooke et al. (1996), the usability score is calculated by summing the contribution (C) of each question (i) and multiplying the result by 2.5. For each odd

numbered question, the contribution is the response minus one; for each even numbered question, the contribution is five minus the response. For the purposes of the experiment, the score represents the perceived usability of a device form factor for completing the tasks. As the AR application was identical on all devices, participants were rating the usability of the device, not the software. As each device symbolised one of three form factor categories discussed in section 3.1, the scores serve to provide a general impression of the differences between them. With this in mind, the variations between scores are more meaningful than the individual scores themselves.

A boxplot of the SUS data is shown in Figure 3.6. Based on the responses, the smartphone recorded the highest mean score of 88.83 ($SD = 7$) followed by the mini tablet with 84.5 ($SD = 14.49$) and the tablet with 77.83 ($SD = 13.19$). Based on the scores, the results confirm H1 with the smartphone achieving the highest mean of the three. Participants were recorded commenting that, compared to the other form factors, the smartphone felt “comfortable”, “familiar”, and “easy to use”, and the SUS score reflects this perception. The low standard deviation relative to the other form factors shows a higher level of continuity between participant ratings. In the pre-test questionnaire, 14 of the 15 participants indicated owning a smartphone, so its SUS score may have been partially influenced (subconsciously or otherwise) by the comments mentioned above.

Analysis of the boxplot reveals the mini tablet fared much closer to the smartphone than the scores suggest. Except for the mini tablet’s outlier—a SUS score of 37.5—both share a very similar distribution with identical first (81.3) and third (93.8) quartiles. Removing the outlier and recalculating a mean for the mini tablet results in an adjusted mean of 87.9, a value much closer to the smartphone. As only two of the participants indicated owning a device befitting this category in the pre-test questionnaire, its overall performance can be considered excellent in the context that it was not a form factor with which the majority of participants had experience. The tablet form factor was the most divisive and recorded the widest distribution of scores. Although 10 of the 15 participants indicated owning such a device, it scored lower than both smartphone and mini tablet

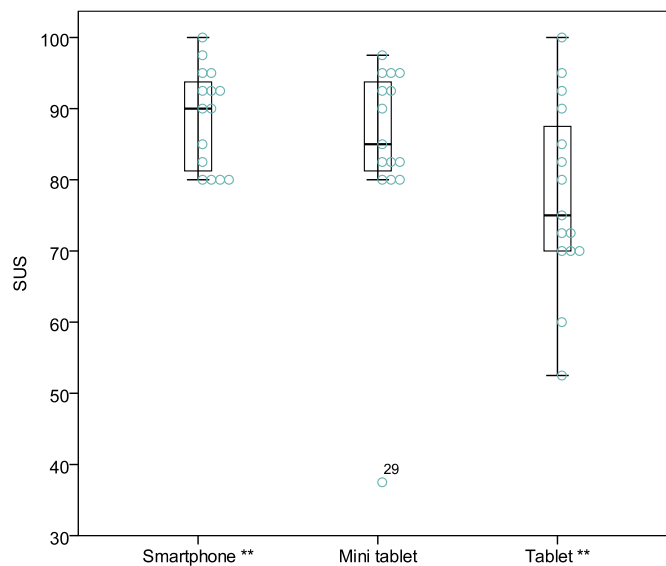


Figure 3.6: SUS data for each form factor. Significance is indicated by ** ($p \leq .01$).

form factors.

To further analyse the effect of SUS means for each form factor, a one-way repeated measures ANOVA was performed. Output revealed a significant difference in SUS score between form factors: $F(2, 28) = 4.73, p = .017$. A Bonferroni post hoc test showed no significant difference between the smartphone and mini tablet ($p = .799$) or between the tablet and mini tablet ($p = .372$). The comparative size difference between these combinations was likely too small to make any discernible usability difference. The difference between smartphone and tablet, on the other hand, was found to be significant ($p = .006$) and indicates the large size gap between these two form factors yields a statistically significant usability difference.

3.6.2 SEQ

Participants provided an SEQ response following the completion of each task. The results are shown in Figure 3.7 as the mean scores for each form factor/task combination.

Both the mini tablet and tablet scored higher than the smartphone for task one suggesting participants found larger devices easier with regards to maintaining device stability and focus over the target. Many comments supported this with remarks including “*a lot more stable*”, “*easier to hold steady*”, and “*better than I thought*”. Unsurprisingly, the increase in screen size was frequently noted as being beneficial to the clarity of the interface and virtual objects. A few participants also stated that they thought the larger devices didn’t emphasise movement as much as the smartphone; however, this interpretation may have been confounded by additional comments related to perceived lag in the smartphone’s camera.

The task two results suggest the effects of device weight becoming noticeable as the lighter smartphone scored 16% better than the heavier tablet; the mini tablet was positioned between the two. Given the task involved users holding and interacting with the device above their head, it is no surprise heavier devices were perceived as being more difficult to use—this would have been further exacerbated by the need to also interact with them. This result supports H3 as the tablet did underperform when compared to the other devices. Many participants commented on the difficulties associated with holding the tablet in a power grip—a grip involving force applied between the palm, thumb, and fingers (Pereira et al., 2013)—with one hand while simultaneously attempting to perform interactions with the other. Given all but one of the participants elected to use a landscape orientation, this would have resulted in the centre of gravity for the device being furthest from their grip. Consequently, terms such as *insecure* and *slipping* were often used when describing the experience.

Both the mini tablet and tablet scored slightly better than the smartphone in task three. This was the first task to involve an interaction beyond a simple tap. It was therefore the first time that participants would potentially encounter occlusion issues with smaller screens. Though all participants completed the task successfully, many noted when using the smartphone that their finger or hand partially obscured their view of the path, leading to a perception of inaccuracy. The same comments were not as prevalent when the larger form factors were used.

The smartphone was expected to perform well in task four given its distinct size

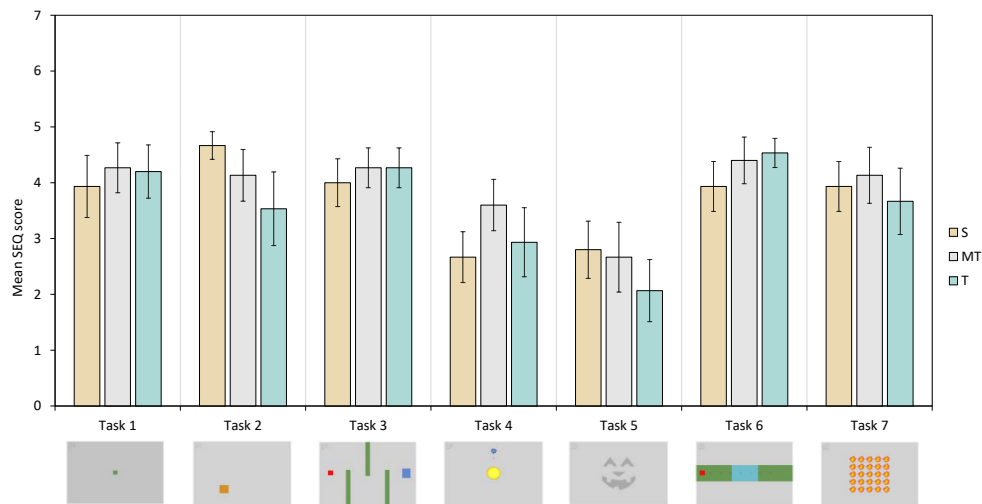


Figure 3.7: Mean SEQ results for each form factor/task combination. Error bars indicate 95% CI.

and weight advantages, but the scores suggest screen size is also an important factor in manoeuvring to maintain target focus. The mini tablet; which balances size, weight, and screen size attributes of the other form factors; bested the tablet by 10% and the smartphone by 13%. Although not originally considered, a side-effect of using a smaller device for physical manoeuvring was the need to make a larger number of adjustments to maintain target focus due to the hand(s) gripping the device being closer to the crosshair in the centre of the screen (the effective pivot point). This subsequently led to increased perceptions of visual tracking latency.

Task five was generally perceived as being difficult on all form factors with none achieving a mean SEQ score greater than three. As the task was located vertically on a wall, participants were required to hold the device out in front of them while interacting. For the tablet, this usually meant adopting a power grip, a ‘dual power grip’ (interaction was performed with the thumbs), or cradling the device from the bottom in the hand’s *vee* resembling the way an easel supports a canvas (Figure 3.8a–c). Conversely, the mini tablet and smartphone were usually held with some form of finger grip in which device support was provided by the fingers only (Figure 3.8d–e). The results show a similar pattern to task two and again suggest weight as being the main issue affecting participants’ perception of task performance. This adds further support to H3. Many who elected to hold the tablet with one hand remarked on the difficulties with which they found securely gripping the device while attempting to draw the face components. In these cases, the hand (extending to the wrist) was frequently observed appearing to strain, resulting in subtle movement of the device the longer the task progressed. [Pereira et al.](#) similarly found large tablets had higher forearm muscle activity and increased wrist extension compared with mid and small tablets when held with one hand. While the tablets they used were non-functional models³, they concluded mid and small tablets were preferable over large tablets due to biomechanical factors, among others. The SEQ results recorded for tasks two and five support this. While the smartphone and mini tablet didn’t exhibit the same levels of strain as the tablet, they still appeared susceptible to a different type of wobble caused by the

³ The large tablet resembled the size of an Apple iPad 2, the mid tablet the size of an Amazon Kindle Fire, and the small tablet the size of a Samsung Galaxy Note.

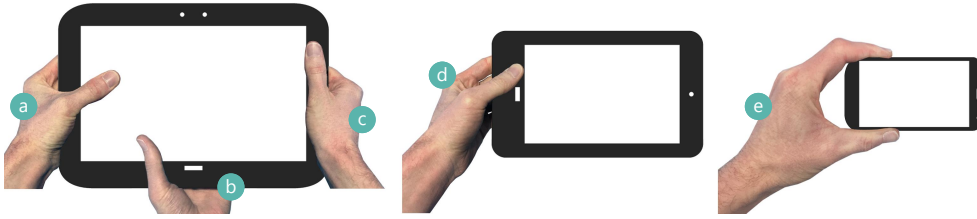


Figure 3.8: Illustration of different device grips observed during the experiment: (a) ‘dual power grip’ (interaction performed with thumbs); (b) cradle grip; (c): power grip; (d/e) finger grip.

counteracting force of the drawing finger moving across the screen surface. Many of these observations could be addressed by implementing some form of assistance whereby the AR scene is paused or ‘frozen’ prior to interaction. This would also negate the suboptimal placement of AR content by allowing the device to be repositioned. Examples of such techniques were discussed in Chapter 2.

Task six involved a similar drawing exercise to task five but in a horizontal orientation at table height. Compared to task five, the results were much improved and projected a mirroring of previous form factor ordering. In a more comfortable orientation, where the device could be cradled from underneath, it appeared that weight did not become the dominant factor and larger screen sizes did have a positive impact on participants’ perceptions of task performance. Remarks to this effect were made by participants with respect to how the task became easier as the device could be more securely supported.

As expected, participants indicated that task seven was repetitive and tedious but not especially difficult. Even with the addition of the TIM tilt interaction, none of the form factors were perceived as being overly straining, but given two hands could be used to carry the device and the interactions were short, weight distribution could be more readily managed. Participants did note the size increase of the larger devices compared to the smartphone with comments such as: “[the mini tablet] does seem a little bit too heavy to hold in one hand” and “[the tablet has] certainly got a bit of weight to it; you can notice a bit of strain and fatigue.” In terms of results, the mini tablet rated first and the tablet last, but all three scored within 7% of each other. As the task involved walking through a public corridor between markers—approximately 30 metres apart—numerous comments were made as participants had time to consider the form factors. Remarks concerning the tablet indicated a perceived awkwardness or social anxiety with its use that may have impacted its score:

- “[Carrying a tablet] feels pretty weird.”
- “It does feel a little weird walking around with a big tablet.”
- “It felt silly holding a tablet like this [arms stretched out] while walking.”
- “I wouldn’t walk around with a tablet if I could help it.”
- “[The tablet] is not really something you would want to carry around.”
- “If feels a bit awkward holding [the tablet] in portrait.”

Post-task reflections suggested the most appropriate form factor to use may ultimately be task and location dependent: smaller devices might be preferable in public settings due to social perceptions associated with the use of larger devices; conversely, larger devices might benefit collaborative settings by allowing several team members to easily share an experience. The *right* device was also suggested to be influenced by personal ergonomics.

What is awkward for one person might be acceptable for another; as one participant pointed out, *“I have big hands so for me a tablet is pretty much a smartphone”*.

However, the data obtained does confirm H3 and indicate weight is an issue that affects the performance of tablet form factors. The SEQ results suggest tablets and mini tablets can be beneficial when neither weight nor sub-optimal task orientation is the overwhelming factor, but smartphones are preferable when they are. The larger screen size of the mini tablet and tablet offered improved scores for both TIM and on-screen interactions when the devices could be gripped securely—for example, with two hands. As shown in Figure 3.7, the only tasks in which the smartphone bests both the mini tablet and tablet are two and five, which, as discussed, are tasks placed in sub-optimal orientations. However, prolonged interaction accuracy is less distinct. Figure 3.9 shows the union of all participant shapes for the face components required in task five. The images are separated by form factor and show renders with and without the template silhouette. The tablet results do exhibit an advantage over the smartphone in terms of shape accuracy. There is more definition to the nose and mouth shapes and overall the composite image more closely represents the pumpkin face. The mini tablet, on the other hand, appears closer to the smartphone than the tablet. While there is still increased definition in the nose shape, the remaining components are less defined—particularly the mouth—and appear equivalent to or worse than the smartphone.

3.6.3 Fatigue

With a reference to the tasks they had just completed, participants provided a subjective rating of fatigue for each combination of form factor/task on a five-point Likert scale with

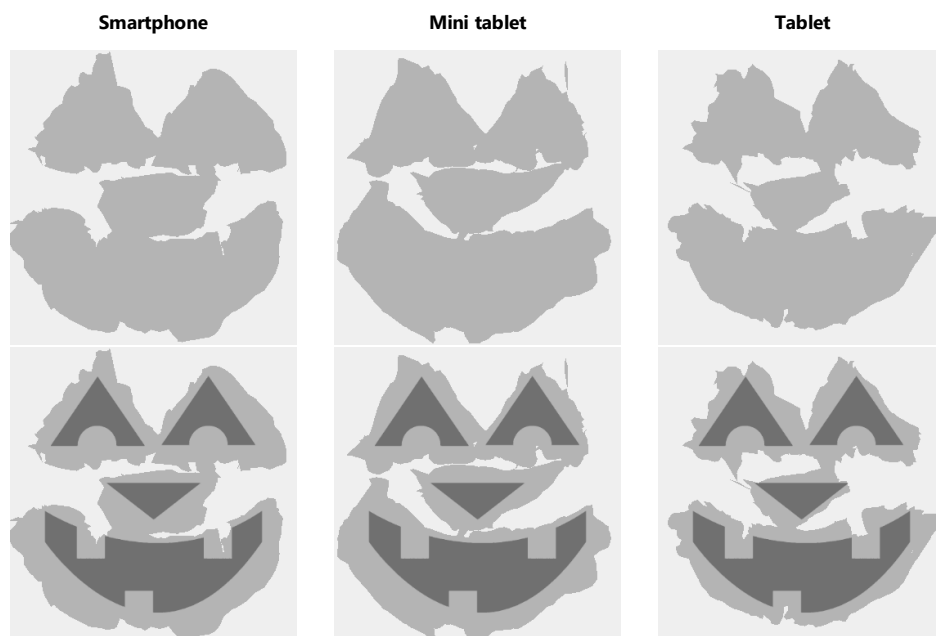


Figure 3.9: Composite render of task five results (by form factor). The top row shows the union of all participant shape components; the bottom row superimposes the template pumpkin face silhouette for reference.

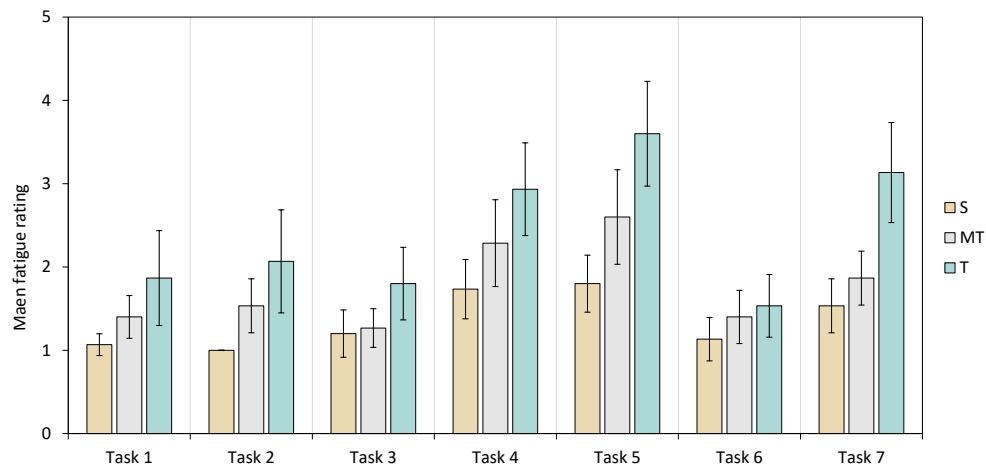


Figure 3.10: Mean subjective rating of fatigue for each combination of form factor/task. Higher values indicate higher levels of perceived fatigue. Error bars indicate 95% CI.

one representing ‘no fatigue’ and five ‘extreme fatigue’. The mean results are presented in Figure 3.10. In each case, the results reflect the ascending size and weight order of the devices: the smallest and lightest form factor, the smartphone, received the lowest fatigue score in all tasks while the largest and heaviest form factor, the tablet, received the highest. Again, the mini tablet was positioned between the two. A Kendall’s tau-b correlation between fatigue and per-task SEQ score reveals a negative relationship for all form factors. As fatigue increased, SEQ score decreased. The tablet showed the largest relationship ($T_b = -.527, p < .001$) followed by the mini tablet ($T_b = -.455, p < .001$) and the smartphone ($T_b = -.404, p < .001$).

With respect to improving device ergonomics and potentially alleviating fatigue levels, some of the commentary recorded during the sessions suggested a grip or handle mechanism would assist with holding larger devices, especially the tablet. These comments mainly occurred during completion of task five—

“Some sort of pad on the back perhaps; something I can grip”

—and seven:

“I feel like I want a handle” and “if the back of the device had an added strap or support that you could hold on to, I think that would be all you’d need to feel comfortable.”

The addition of such ergonomic improvements to tablets has been explored previously with positive results. [Pereira et al.](#) found the addition of a ledge or handle grip to be beneficial over a flat grip of the device bezel. With the former options, they recorded reduced ratings of fatigue and an increase in grip security, although they do note the results may have been confounded by differences in tablet weights. [Veas and Kruijffs \(2008\)](#) Vesp’R, discussed previously in Chapter 2, similarly showed good levels of ergonomic and user comfort by distributing device weight evenly via the use of bespoke flanking handles. As technology continues to advance, the natural evolution of tablet devices may contribute to improving ergonomics through reductions in thickness, weight, and physical design modifications.

As mentioned in section 3.6.2, an additional solution to managing the use of larger devices for AR, and subsequently mitigating fatigue, is the use of a pause or freeze technique to enable the device to be temporarily repositioned to a more comfortable, non-fatiguing pose. While this does not negate the weight of the device, it does allow it to be managed by users so it doesn't dominate the user experience.

3.6.4 Subjective Responses

As a point of interest in the study, and to gain an insight into participant perceptions, participants were asked to rate how important they believed certain hardware factors would be for handheld AR use. The question was asked twice: first in the pre-test questionnaire before any devices were used and again in the post-test questionnaire. Answers for each factor were given on a five-point Likert scale with one representing 'extremely unimportant' and five 'extremely important'. The results from each question are provided in Figure 3.11 while a ranking of the factors is given in Table 3.2.

Surprisingly, participants ranked *weight* last in the pre-test ranking even with full knowledge of the tasks and form factors they would be using to attempt them. Once the tasks were complete, however, *weight* moved to second place behind *ergonomics*, likely in response to the experience of using the larger devices—the tablet in particular—and the discussion of H3. It could be argued that the initial low ranking of weight was mainly due to the way participants typically used their own larger handheld devices. Although 10 of the 15 indicated tablet ownership in the pre-test questionnaire, only one of those

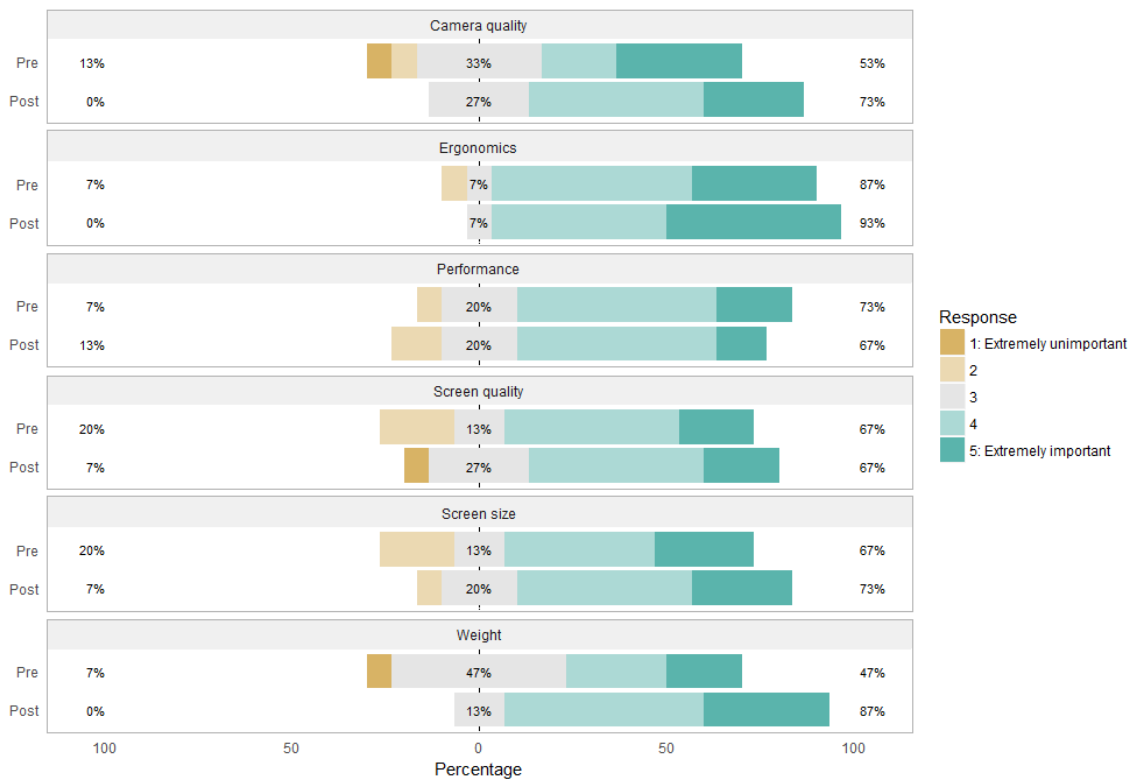


Figure 3.11: Comparison between pre- and post-test rating of hardware factors believed to be important for handheld AR.

Table 3.2: Participant ranking of hardware factors believed to be important for mobile AR.

RANK	FACTOR	
	Pre-test	Post-test
1	Ergonomics	Ergonomics
2	Performance	Weight
3	Screen size	Camera quality
4	Camera quality	Screen size
5	Screen quality	Screen quality
6	Weight	Performance

responded *yes* to a follow-up question asking whether they used the device outside of their home. In this setting, it is reasonable to assume the primary engagement with the device⁴ would involve it sitting in a lap or resting on a surface—all use cases in which weight is unlikely to be a noticeable factor. In fact, one participant made a remark to this effect during one of the tasks: “*I’m usually sitting down when using a tablet.*” AR obviously presents different demands. When faced with a situation in which the device is required to be continually held up, its weight would suddenly become much more apparent. This would explain the *weight* attribute’s position shift. The swap with *performance* was not considered an indication that it is unimportant but rather an acknowledgement of the current state of mobile hardware as being sufficient for the purposes of general handheld AR use. As all three of the devices used for the experiment provided adequate performance for the developed application, *performance* was unlikely to have been a focal point when participants considered their post-test ranking.

The other ranking change was between *screen size* and *camera quality*, although these only shifted one place. As mentioned in section 3.6.2, several participants commented that the smartphone camera appeared to present noticeable lag. It is likely this position change was a result of such an experience as well as confounding hardware characteristics of the devices used. With this in mind, it was probable participants considered it more important that images remain responsive rather than appear larger, which would be associated with *camera quality*. Another factor that may have contributed to the change was the observation (by participants) regarding heat build-up around the devices’ camera area after they had been used for upwards of ten minutes—the mean usage time for each device to complete the tasks. The off-the-shelf hardware used was evidently not able to dissipate heat build-up quickly enough, causing a hot-spot around the camera to manifest. While these devices were unlikely to have been engineered with continual camera use in mind, it is an aspect of hardware design that affects prolonged AR use. Thermal dissipation was previously noted by Höllerer and Feiner (2004) as a challenge of small high-performance mobile AR systems.

In addition to the importance of various hardware factors, participants were also asked for a subjective form factor preference, both before and after the test in the respective pre- and post- questionnaires. The question was framed in terms of which form factor they would choose for completing the tasks—either for the first time or again—if they could only pick one. Responses are reported in Table 3.3. Most participants elected not to

⁴ Such as reading an email, browsing the Internet, replying to a text message, checking social media, or playing a game.

Table 3.3: Participants' subjective preferred form factor

PARTICIPANT	PRE-TEST	POST-TEST	
1	S	MT	*
2	MT	S	*
3	MT	MT	
4	T	MT	*
5	MT	MT	
6	T	MT	*
7	MT	MT	
8	MT	S	*
9	S	S	
10	MT	T	*
11	MT	MT	
12	MT	MT	
13	S	S	
14	S	S	
15	S	MT	*

change their preference between pre- and post-test responses. Those that did (indicated with an *) always selected the next adjacent form factor; for example, smartphone to mini tablet or tablet to mini tablet. There were no cases where participants' responses changed from smartphone to tablet or vice versa. The mini tablet was the most subjectively preferred form factor in both cases and consistently performed well in all analyses, placing either first or second; it never performed worst. Although the SEQ results and participant comments propose the 'right' form factor as being dependent on use case and personal ergonomics, given a choice of only one device, participants elected to choose the form factor that offered the best compromise between the most desirable attributes of both the smartphone (size and weight) and the tablet (screen size). This result rejects H2.

“[The mini tablet] is a nice compromise. I quite like this compromise.”

—Participant 6

3.7 Summary

This chapter has explored the effects of handheld device form factor on AR usability. Participants were asked to complete seven interactive tasks involving TIM and touch interactions using a smartphone, mini tablet, and tablet device. Tasks evaluated factors including stability, interaction accuracy, occlusion, position and orientation effects, manoeuvrability, device weight, and fatigue. Participants provided feedback on form factor performance and usability following each task (via the SEQ) and the use of each device (via the SUS). They were also recorded during their task attempts and encouraged to provide verbal feedback on perceived differences between form factors as they progressed.

The smartphone recorded the highest overall SUS rating and received the greatest continuity between participants' scores. The mini tablet performed close to the smartphone once its results had been normalised, while the tablet was the most divisive and rated the least usable. Per task performance varied between form factors based on orientation, required accuracy, occlusion, weight, and subjective preference; however, the mini tablet always rated first or second. Fatigue results reflected the ascending size/weight order of

the devices. The mini tablet was the most subjectively preferred form factor and provided the best overall experience.

Results showed larger tablet form factors continue to present issues for handheld AR use based on their intrinsic size and weight characteristics. Holding such devices in poses where only the hands and arms can support the device was observed to be straining and lead to underperformance—this was especially apparent for tasks requiring simultaneous interaction. However, tablet devices were suggested to offer improvements in task easiness, performance satisfaction, and interaction accuracy so long as weight or sub-optimal device pose were not overwhelming factors.

The following chapter incorporates the conclusions derived from this chapter and reports on the design and development of a handheld AR authoring tool for inexperienced users.

Handheld Augmented Reality Authoring Tool *for Inexperienced Operators*

This chapter describes HARATIO, a Handheld AR Authoring Tool for Inexperienced Operators. The tool has been created to assist with evaluating user interfaces and user experiences suitable for novices unfamiliar with technical concepts such as AR, programming, and 3D modelling. To support evaluation across a variety of device form factors, HARATIO has been developed using an architecture leveraging existing platforms and frameworks. The overall design prioritises simplicity and coherence while also incorporating usability findings from the experiment reported previously in Chapter 3. The development of HARATIO's interface has been guided by established usability heuristics for user interfaces (Nielsen, 1994). In the sections that follow, detailed descriptions of the interface and interaction experience are presented along with discussion of existing work that has informed many of the design choices.

Authoring tasks within HARATIO are separated into three main activities that correspond to creating scene content, editing scene content, and defining interactivity for objects within a scene. Activities are supported by a central menu interface that adapts a typical radial menu design to provide a screen efficient and distraction-free presentation. Object interactivity is defined using a simplified script editor that implements a dynamic visual drag-and-drop interface and assumes no programming experience. The editor includes enhanced support to novices in the form of functionality that translates scripts into natural language sentences. The menu and script editor comprise key user interface components of HARATIO and represent aspects of the tool users will interact with most while authoring AR content.

The description of HARATIO throughout this chapter represents the tool in its final evolution to provide a complete discussion of all functionality. Iterative improvements leading to this version were made based on empirical data and user feedback collected from evaluations discussed over the succeeding two chapters. Changes made in response to the evaluation reported in Chapter 5 are covered at the beginning of Chapter 6.

4.1 Architecture

HARATIO was developed to support the investigation of interfaces suitable for mobile devices, AR authoring, and novice users. As such, existing platforms and frameworks were leveraged where appropriate to provide the foundation on which to explore these goals. This section discusses the use of Unity and Vuforia and describes how each has been integrated into the overall design.

4.1.1 Unity

HARATIO was initially intended to be a native Android application leveraging Vuforia's Android Software Development Kit (SDK) for tracking functionality. Early in the development process, however, this was abandoned in favour of adopting Unity (version 4.6.7f1; [Unity Technologies SF, 2014a](#)) as a middleware platform. Although Unity is primarily a game engine (with an accompanying eponymous development environment), it includes many features well-suited to prototyping a tool such as HARATIO, and it was used successfully to create the tasks for the user evaluation discussed in Chapter 3.

In terms of developing an AR authoring tool, Unity provides much of the groundwork that would otherwise need to be developed. Tasks common to working with a 3D environment, such as physics and rendering operations, are built in, and features not included in default libraries are often available as additional plugins. The other major strength of Unity is its cross-platform support. Although Android was the intended target for evaluation, deployment profiles are available for all major mobile and PC platforms—to deploy HARATIO to another platform, all that is required is to select the appropriate profile from within Unity and recompile. Minor adjustments may be needed to accommodate differences in the *new* platform, but the underlying capability this provides allows a tool such as HARATIO to be deployed with a quasi 'hybrid interface' ([Feiner and Shamash, 1991](#)) where the end device and interaction mechanisms can be selected based on what best suits the current environment or task. In some cases, this might be a handheld device; in others, it might be a desktop PC. Ultimately, Unity offered the kind of flexibility that allowed development time to be focussed on user-facing aspects of HARATIO that were more pertinent to the goals of this research and the creation of a novice-friendly authoring tool.

Everything in an application developed with Unity is fundamentally represented as a `GameObject`—this includes lighting, cameras, asset instances, and user interface elements, among others. Game objects are synonymous with scene objects and additionally act as containers for `Components`, which are used to describe an object's state and behaviour. The two share a many-to-many relationship; that is, a game object can contain multiple components and a component can belong to multiple game object instances. Example components include `(Rect)Transform`, which provides details on the location of a game object within a scene (in world or screen coordinates), and `Light`, which describes the properties of a light source. Components may also be scripts. These enable custom functionality and are written in either `C#` or `UnityScript`¹ on top of the Mono runtime. The modular architecture afforded by Unity via game objects and components provides considerable flexibility with regards to designing and developing applications. HARATIO is predominately implemented as a series of custom scripts (components) attached to various game objects.

A high level architecture diagram is presented in Figure 4.1. The diagram illustrates the layers that comprise the system, starting from the physical device and host operating system through to the use of Unity as middleware and the HARATIO application itself. The diagram also depicts two additional plugins that have been added to Unity to extend the platform and aid in HARATIO's development. Vuforia, covered in the following subsection,

¹ UnityScript is a Unity-specific language modelled on JavaScript.

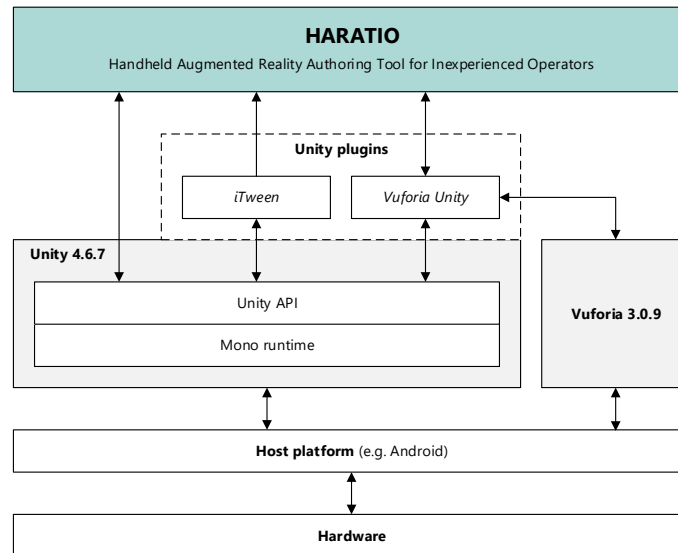


Figure 4.1: Architectural overview of HARATIO

provides AR tracking capabilities necessary for the development of an AR application. The iTween plugin (version 2.0.46; Berkebile, 2011) provides an interpolation-based animation framework. This has been used to create fluid motion effects within the user interface and assist with Behaviour script implementation (discussed in section 4.4.5).

4.1.2 Vuforia

As the version of Unity used did not natively support AR, an additional plugin was required to provide tracking and registration functionality. HARATIO utilises the Vuforia framework (version 3.0.9; Qualcomm Connected Experiences Inc., 2014) for this purpose. At the time development commenced, Vuforia’s tracking capabilities consisted of fiducial and feature-based approaches.

Fiducial markers have long been used for AR tracking and are well-suited to mobile devices where resources are limited (compared with desktop systems) and must be managed accordingly. Vuforia’s implementation of fiducials varies from the traditional symbol-based markers commonly used with popular toolkits like ARToolkit (Kato and Billinghurst, 1999). Symbol-based markers rely on abstract black-and-white images that the tracker matches against a template database to determine the corresponding marker ID. The approach is subject to limitations in practical library size—computation rises with library size as each marker must be evaluated for a match—and the occurrence of false-positives. In response to these limitations, Fiala (2005a) described an alternate design in which the marker ID was encoded as a binary pattern in place of an image. Along similar lines, Wagner, Langlotz and Schmalstieg (2008) proposed the *frame marker* design that encodes the ID as a binary pattern within the marker’s border. Vuforia’s implementation of fiducial markers utilises frame markers. The border pattern supports 512 unique combinations, all of which may be used in a single application. Frame markers within Vuforia are recommended to be between three and ten centimetres in size; the chosen size affects the scale of objects registered to the marker (discussed in section 4.4.3.1). For simplicity, the framework suggests a base reference translation of one marker unit to one scene unit be used.



Figure 4.2: Example of default (left) and customised (right) frame marker designs.

By utilising the marker's border to encode the ID, the interior area of a frame marker is left free to customise with images or text specific to the marker's intended use. Figure 4.2 shows an example of a customised HARATIO marker in comparison to a default Vuforia one. The interior has been customised to include a brief description for how the marker should be used.

Vuforia's integration with Unity consists of a series of specialised components that comprise the plugin. These components enable standard Unity game objects to be enhanced with AR functionality and communicate with the underlying Vuforia SDK. Components added to a Unity Camera object, for example, enable it to render the video background from the physical device camera as well as manipulate other game objects based on tracker input. In this configuration, a Unity camera is used to represent the user's viewpoint of the scene. Each object Vuforia tracks is represented by a `Trackable` component attached to a corresponding game object. As HARATIO utilises frame markers for tracking, a game object corresponding to each physical marker is used along with a frame marker component that identifies its ID. Virtual objects associated with the marker are then added as children and adopt a relative position within the scene. The Vuforia tracker will update the position and rotation of the parent marker object in response to pose changes to ensure it stays registered with the corresponding physical fiducial. All child virtual objects will also be updated because of the parent-child relationship. The visibility of child objects is tied to the trackable state of the marker, and the Vuforia tracker will also update this accordingly.

4.2 Radial Menu

The radial menu provides access to all of HARATIO's modes, activities, and operations. It has been designed to be simple to operate and utilise screen space efficiently. The menu supports a multi-level hierarchy and displays child menus as separate concentric rings. Only the outermost ring (the deepest child menu) is fully visible with parent menus appearing in a semi-collapsed state. This is intended to minimise screen space use as well as focus the user's attention on those menu options relevant to the current task. The centre of the menu facilitates moving between AR (section 4.3) and author (section 4.4) modes as well as accessing parent menus; it can also display supplementary information for specific operations, such as object volume when performing a scale. To avoid occluding other interface elements or underlying AR scene objects, the menu is moveable and may be repositioned to any location on the screen. The remainder of this section discusses the evolution of HARATIO's menu based on existing designs and informal feedback.

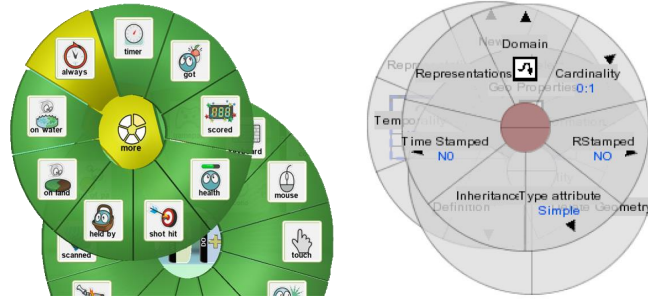


Figure 4.3: Overlapping pie menu designs. Kodu (MacLaurin, 2011) presents child menus centred over the invoking item (left). Rubio and Janecek (2002) explored transparent child menus (right).

4.2.1 Background and Inspiration

Linear menus are commonly used in graphical user interfaces to display groups of related options in vertical or horizontal lists. The mobile AR modelling tool presented by Tang et al. (2015) employs multiple linear menus to facilitate access to various modelling and editing options. Their implementation allows the menu to be docked to either left or right screen edge to better support the preferred handedness of the user. Although linear menus are well-suited to exposing many options, they can quickly encroach on available screen space when displaying multi-level hierarchies. For mobile devices, this makes it difficult to display larger menu systems while keeping menu item sizes appropriate for touch interaction, especially on smaller screens (Francone et al., 2010). While linear menus may be made scrollable, this affects discoverability of off screen (overflow) items.

A common alternative to a linear menu is a pie menu, which displays items around the circumference of a circle equidistant from its centre. Compared to a linear menu, this approach affords increased target size and fixed target distance (Callahan et al., 1988), which has benefits in item acquisition time as per Fitts' law (Fitts, 1954). However, pie menus only natively support a single menu level and are practically limited in the number of items they can contain. To support hierarchical structures, additional pie menus can either be presented in overlapping or non-overlapping states. Kodu, a visual programming editor discussed in section 4.4.5.1, utilises overlapping pie menus for this purpose. The editor displays up to nine options in a single menu and offers a tenth to access further options in an overlapping menu. The overlapping menu is spawned centred on the item that invoked it. While this maintains a fixed target distance to items in the *new* menu, it results in an increase to screen space consumption—non-overlapping menus are even worse in this regard. The relationship between subsequent menus can also become obfuscated by the act of each child menu partially or fully occluding its parent. Rubio and Janecek (2002) added transparency to their overlapping pie menu implementation with the aim of providing a view through the menu to occluded items beneath. However, this quickly results in the clarity of the menu being adversely affected by bleed-through of underlying content. Through observation, both Rubio and Janecek concede that users may confuse commands between menus when the opacity is set too low. Both designs are shown in Figure 4.3.

In the context of HARATIO's overall design goals and target device type, pie menus were considered preferable over linear menus; however, in terms of supporting a menu

hierarchy, both overlapping and non-overlapping pie menus were considered an inelegant solution that results in unnecessary screen clutter. Alternate approaches to supporting menu hierarchies that does not involve separate pie menus typically incorporates child menus as inner or outer concentric rings that appear adjacent to their parent. The Wave menu (Bailly et al., 2007), a variation of marking menus (Kurtenbach and Buxton, 1994), displays each child menu at its centre with parent menus expanding outward upon the user drawing a stroke gesture (Figure 4.4a). The outermost ring therefore represents the root menu while the centre represents the deepest child menu traversed. Although the Wave menu increases in size with each child menu traversal, this approach allows the menu to remain usable even in situations where parts of it may be clipped off screen. The Wavelet menu (Francone et al., 2010) extends the Wave menu for mobile platforms and incorporates a touch gesture interaction model. To access child menus, users drag their finger from an empty central area towards the desired menu option (Figure 4.4b). This causes the menu to animate outwards revealing the child menu beneath. The authors describe the approach as utilising a stacking metaphor. Like the Wave menu, each parent menu in the hierarchy appears as a concentric ring adjacent to its child with the outermost ring representing the root. Tapping the central area collapses the deepest child menu while double-tapping collapses all child menus. Both Wave and Wavelet designs support pre-visualisation by allowing the user to explore child menus without making a selection. Contrary to Wave and Wavelet designs, the Compact Radial Layout (CRL) (Samp and Decker, 2010) presents child menus as outer surrounding rings that are tightly packed with their parent (Figure 4.4c). Menu items leading to child menus are indicated with a line, and selection of one of these causes the associated menu to appear. Items not marked with a line instead perform actions when selected and result in all child menus disappearing. The CRL maintains positional consistency (Somberg, 1987) between already selected items (unlike Wave and Wavelet) and also preserves the context of the user's path through the menu.

The menu design used in early versions of Microsoft's 3D Builder (Microsoft Corporation, 2014), a tool designed for easily customising and printing 3D models, shares similarities to the CRL by adopting a tightly-packed, layered design that incorporates outwardly-expanding rings for child menus. The menu, shown in Figure 4.4d, prioritises compactness and displays menu items as icons sans label. Like the CRL, selection of a menu item will either invoke a corresponding action or cause a child menu to appear as an outer surrounding ring. Child menus are displayed in different colours based on application mode, and the hierarchy is limited to two levels: the root and one child menu. The menu design also makes efficient use of the central area by displaying context-sensitive information related to the current task. The figure graphic shows object translation enabled—indicated by the light blue colouring of its icon—and the centre of the menu exhibiting a corresponding indication of distance moved in millimetres. The use of colour and context-sensitive information adds to the level of feedback provided by the menu.

The CRL and 3D Builder menus offered significant early inspiration for HARATIO's menu direction. The latter, especially, was used as a starting point for exploring menu layouts based on its implementation of 3D modelling features. Figure 4.5 shows a series of proof-of-concept illustrations for a prototype HARATIO menu design utilising the 3D Builder layout and demonstrates the transition sequence between different menu levels.

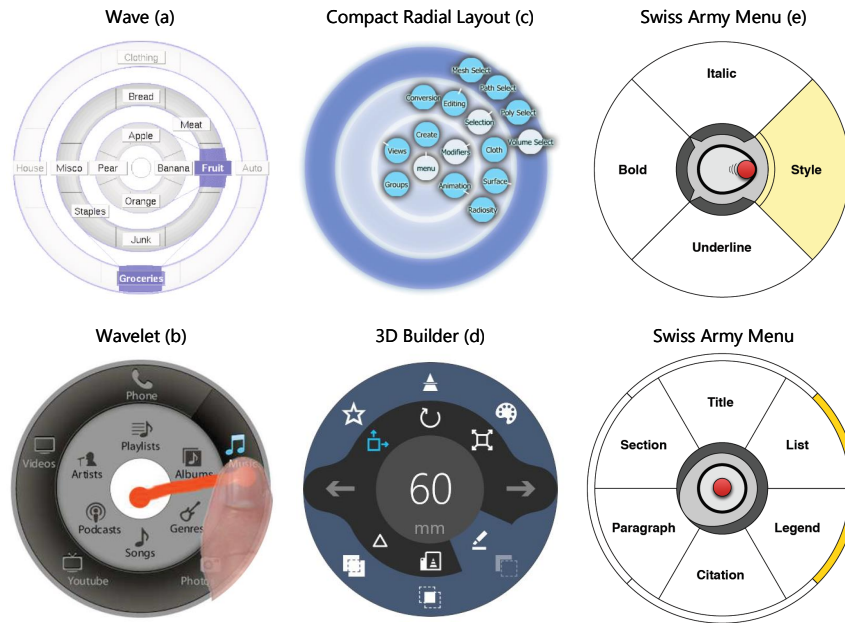


Figure 4.4: Non-overlapping pie menu designs. (a) Wave menu (Bailly et al., 2007); (b) Wavelet menu (Francone et al., 2010); (c) Compact Radial Layout (Samp and Decker, 2010); (d) 3D Builder menu (Microsoft Corporation, 2014); (e) Swiss Army Menu (Bonnet and Appert, 2011)

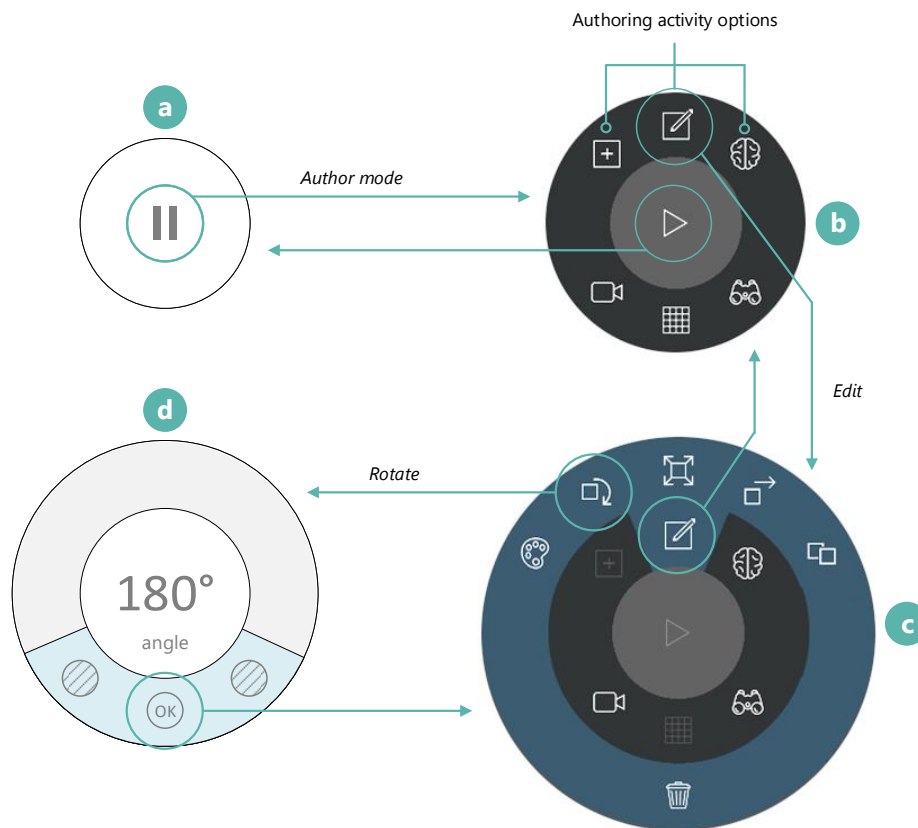


Figure 4.5: Initial HARATIO menu concept based on Microsoft 3D Builder layout. Images on the left represent initial design illustrations. Images on the right represent renders produced from illustrations.

The rightmost two graphics in the figure depict a rendering of their adjacent illustration (as indicated by the arrows).

The menu is initially configured in a collapsed state when viewing live AR content. This is represented as a circular area with no visible surrounding menu content (a). The circle contains a single button to facilitate switching between HARATIO's AR and author modes. Selecting this button reveals the root menu as a concentric ring. This menu serves as the launching point for authoring activities and provides access to viewport options (b). As authoring activities represent the most important aspects of the tool, they are placed at the top of the root menu. Samp and Decker (2011) suggest placing important or most frequently used items at the top of the first menu as the visual search pattern starts from the twelve o'clock position; such items will subsequently be found faster. Menu items function as per the CRL and 3D Builder designs and either invoke an operation or reveal a child menu. Selection of the `Edit` button, for example, displays a child menu with Edit activity operations (c). In certain circumstances, the menu can deviate from this behaviour and completely change appearance to focus the user's attention on items specific to a certain operation, such as rotate (d), scale, or translate. In the case of (d), the centre of the menu displays information related to the operation while the surrounding menu items provide operational controls; selection of the `OK` button returns to the previous menu configuration (c), and reselecting the `Edit` button collapses the child menu and returns to (b). The menu can be repositioned in any of these states to suit the user's grip of the device, personal preference, or to avoid scene occlusion. Though this proof-of-concept represented a good starting point, a number of usability concerns were identified with the design that resulted in a re-evaluation of certain elements:

- It was not clear or obvious what menu items corresponded to as they were represented by icons only.

Wiedenbeck (1999) discovered icons with text labels or text labels alone to be beneficial to new users. Leung et al. (2011) noted similar results when evaluating the differences between younger and older adults for mobile device icon usability. Both suggest labelling icons, at least initially. Given the target audience of HARATIO, displaying menu items without labels would adversely affect discoverability and likely lead to confusion and frustration.

- The compactness of the design meant menu item distribution was dense, both in terms of the distance between items within the same menu as well as between adjacent menu levels. Menu items were therefore considered prone to miss-selection due to smaller hit targets and the inaccuracies of touch interaction. This would be exacerbated on smaller screens.

Research has shown touch interaction using a finger to be imprecise, especially when selecting smaller targets (Cockburn et al., 2012). Inaccuracies are commonly attributed to a combination of ambiguous finger contact point and occlusion—the so-called fat finger problem (Vogel and Baudisch, 2007). The addition of icon labels would enable a corresponding increase to the size of each menu item and would also enlarge the size of each menu ring, providing more space for dispersing items. In contrast to the arrangement of menu items

in the second menu level of Figure 4.5c, the 3D Builder example provided in Figure 4.4d shows both visible menus containing an equal number of items and adopting a ‘SPARSE’ layout (Samp and Decker, 2010). Modifying the layout of menu rings to similarly fix the number of items visible would also help. These changes would, however, negatively affect the compactness of the design.

- As child menus expanded outward, larger hierarchies would potentially suffer from clipping on smaller screens, making interaction more difficult. The presentation of such hierarchies could also be overwhelming for novice users given the number of menu items potentially visible at once.
- Changing menu presentation, as in Figure 4.5d, affects operational consistency where selection of a menu item would normally be expected to invoke an action or reveal a child menu as a surrounding ring.

The Swiss Army Menu (SAM) (Bonnet and Appert, 2011), a radial menu designed for single-handed use on small mobile devices, offers an alternate approach to supporting larger menu hierarchies while also maintaining relative screen space efficiency. The SAM design itself is inspired by the Wavelet menu discussed previously and utilises the central area in a novel way to facilitate indirect menu navigation and operation. Users manipulate a small ‘puck’ (a red dot) in conjunction with a surrounding ‘rubber band’ to access child menus or interact with menu items. Like the Wave and Wavelet menus, SAM reveals child menus as inner concentric rings when invoked with the puck (Figure 4.4e). Parents of the active child menu are collapsed to a slim border with a persistent highlight indicating the relative location of menu items that led to their selection. Incorporating such an approach into HARATIO’s menu design would allow for adequate menu item sizes and improved screen space efficiency while also supporting larger hierarchies. It would also enable an indication of selection history to be elegantly displayed for each invoked menu item, which would serve as a *visual breadcrumb* of the user’s path through the menu hierarchy that preserves positional consistency.

The design of the menu was subsequently modified to address these concerns. As indicated by the above remarks, the modifications primarily restructured the menu to incorporate item labels and provide an improved presentation of child menus. The final structure of the menu is covered in the following subsection.

4.2.2 Final Structure

HARATIO’s final menu design is illustrated in Figure 4.6. The structure is an adaptation of a pie menu that supports multi-menu hierarchies via a radial design. The menu is composed of three main elements, each of which are illustrated in the exploded view given in Figure 4.7. The elements comprise the menu core, collapsed (inactive) menus, and the current active menu.

Each active menu is limited to displaying eight standard menu controls. A standard menu control refers to a button that either invokes an action or spawns a child menu. In

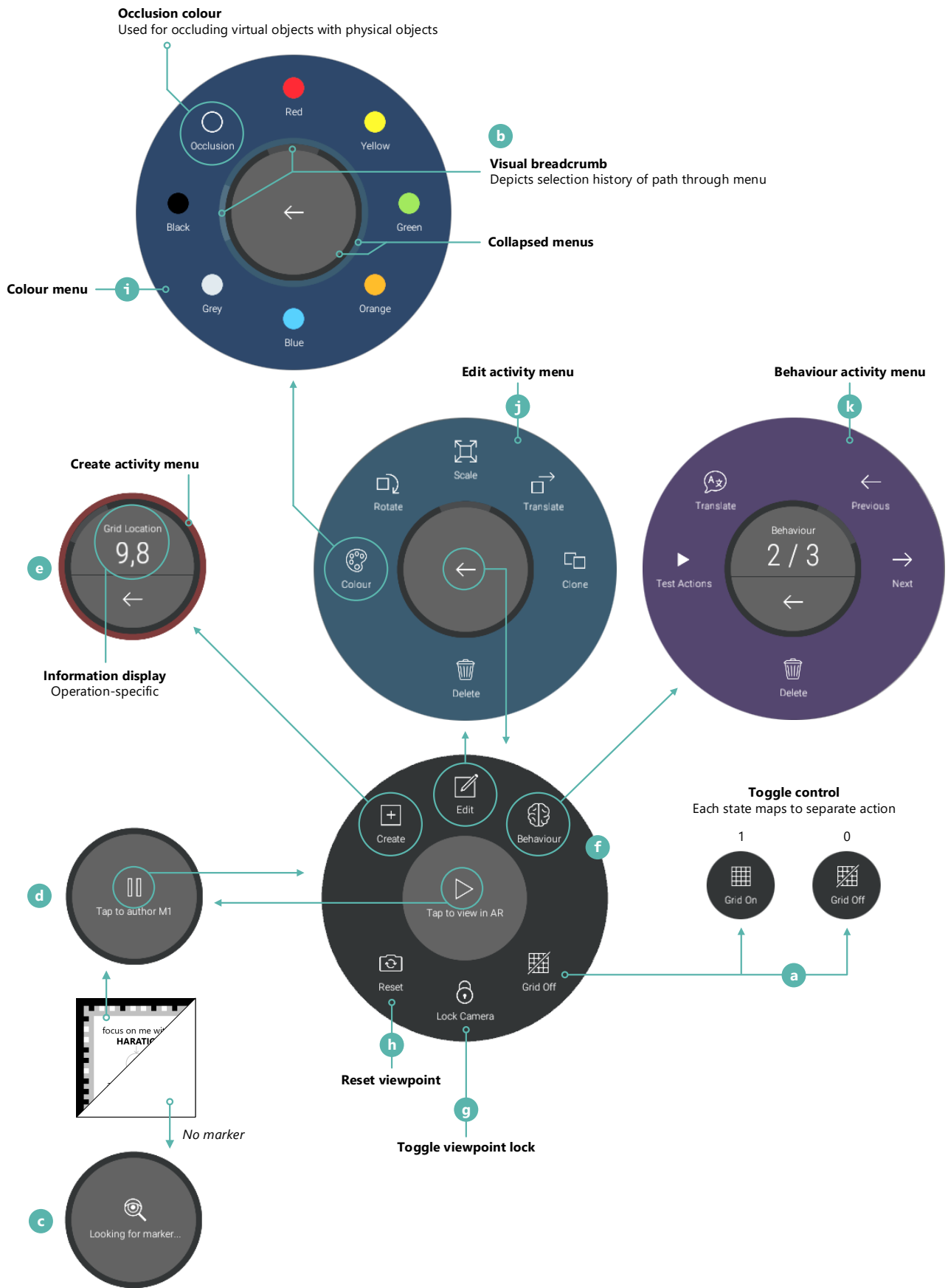


Figure 4.6: Overview of HARATIO's final radial menu design

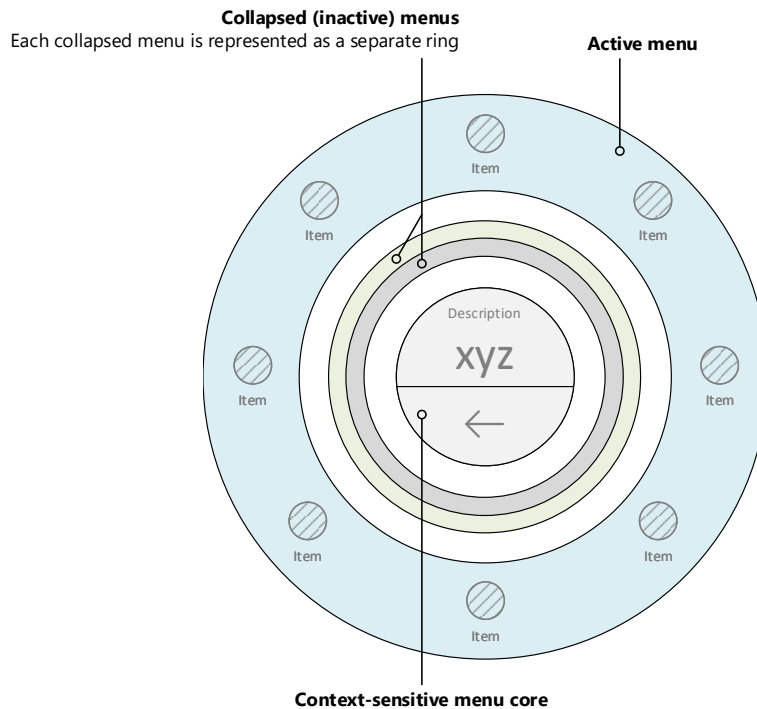


Figure 4.7: Exploded view of radial menu components

in addition to their iconographic representation², buttons are also displayed with supporting labels. As noted previously, the literature on various icon presentations has found labels to be beneficial to new users of a system. Given HARATIO's target audience, the inclusion of button labels seems entirely appropriate. With the addition of labels and limits on menu density, target size for each standard control has been increased by a factor of three over the initial design. This provides considerably more space around buttons to better support touch interaction. Standard menu controls also maintain positional consistency within their respective menu level and appear disabled rather than hidden in cases where they are inapplicable. As [Somberg \(1987\)](#) remarks, users quickly become accustomed to the positioning of items within menus and rely more on positional consistency as they become familiar with the system.

Additional menu controls have also been explored. Toggle controls are a specialised type of standard control that function like a checkbox and switch between binary states when selected. Each state may map to a separate action to enable different functionality as well as include an alternate icon and label. Figure 4.6a shows an example of a toggle control used to manage grid visibility (section 4.4.3.1).

The rotate and scale operations within the Edit activity utilise a custom slider control to facilitate the discreet adjustment of values. This control consists of a track that occupies all or part of the active menu space as well as a *handle* (affixed to the track) that can be manipulated by the user to adjust values. The circular nature of the radial menu lends itself well to presenting a slider control in this format. The implementation of the slider

² Icons used throughout HARATIO were sourced from Icons8 (Icons8 LLC, 2017), a website offering icon designs with a license model appropriate for application prototyping. Some icons have been modified to better fit within the design where an existing option wasn't available. As per the license, an attribute appears within the tool to acknowledge the icons' source.

control is covered further in the respective discussion of rotate and scale operations in section 4.4.4.2.

The core of the menu is persistent and adapts its content automatically based on the current state of the menu, application, or task. It is capable of housing a configurable label, a single button, or a unique split view combining both. The label-only configuration displays simple non-actionable messages (Figure 4.6c). When using HARATIO in AR mode, this configuration is activated whenever a fiducial marker is not being actively tracked. The button configuration provides a single button for toggling between AR and author modes or navigating back through the menu hierarchy (Figure 4.6d). The determination as to which is displayed is based on the current mode and number of child menus traversed. The final configuration separates the core into two sections (Figure 4.6e). The top half displays additional information related to certain operations, maintaining the functionality inspired by the 3D Builder design discussed in the proof-of-concept; example information includes the size of an object being scaled, the location of an object within a scene, or the object behaviour script being edited. In this configuration, the bottom half always contains a back navigation button. This configuration is only applicable within child menus corresponding to certain activities or operations.

Like the Wavelet design, the menu structure adopts the use of a stacking metaphor. Each menu can exist in either *collapsed* or *active* states, but only one can be active—and therefore visible—at any one time. Collapsed menus appear as rings surrounding the core and communicate the number of menu levels traversed. This is analogous to the way the rings of a tree reveal its age. In the example illustration provided in Figure 4.7, the menu hierarchy has been traversed to the third level; consequently, there are two collapsed menus and one active menu. Initially, when HARATIO is in AR mode, no menus have been traversed and the root menu appears in a collapsed state as shown in Figure 4.6 (c–d). Entering author mode causes the root menu to become active, and it does so by animating outwards from the core.

Activating a child menu results in a similar sequence and can be thought of as ‘pushing’ the child menu onto the menu stack. When this occurs, the parent menu will simultaneously collapse to a ring state in an animation that is the reverse of that used to reveal it. This approach was inspired by the SAM as it helps keep overall menu size compact and space efficient. It also reduces interface clutter by focussing the user’s attention on menu items relevant to the current authoring task without overloading them with other, potentially inapplicable, options. [Tapia and Kurtenbach \(1995\)](#) previously proposed hiding parent menus to reduce interface clutter when describing design refinements to marking menus. Viewing a child menu three levels deep therefore results in the menu core being surrounded by two collapsed menu rings and the expanded active child menu. The size difference between displaying just the root menu verses displaying three levels deep into the hierarchy is ultimately negligible and helps mitigate occurrences of menu clipping on devices with smaller screens.

[Samp and Decker \(2011\)](#) discovered visual search time of items in second and third menu levels to be a function of distance from parent items when all levels were visible. When searching for a menu item, users were observed to alternate between searching both sides of a menu ring, starting from the item closest to the parent (selected item). For the root level, where all items are equidistant from the centre, searching started from the

twelve o'clock position, as mentioned previously. By collapsing HARATIO's parent menus and only fully showing the active menu, users are never presented with more than one menu's worth of items to scan. As each menu is limited to eight items, visual search time therefore remains consistent regardless of menu depth.

Accessing collapsed parent menus (navigating back through the hierarchy) is accomplished using the menu core. When at least one menu is collapsed, the core will provide a back button that returns the outermost collapsed menu to active. This process may also be triggered automatically by certain menu items if their containing menu is modal and requires no further interaction once a selection has occurred—this is referred to as an *auto-collapsing* menu. Reactivating a collapsed parent menu can be thought of as 'popping' the active menu from the menu stack and may be repeated until there are no more collapsed menus and the root menu is once again active.

Menus are displayed in separate colours based on the authoring activities (covered in section 4.4) they correspond to or the operations they support. As with the 3D Builder design, the use of colour helps distinguish menu rings when displayed adjacent to one another and is particularly beneficial to the appearance of collapsed menus. Once users become aware of which colour maps to which activity/operation, they gain additional peripheral awareness of HARATIO's current state without having to study the active menu configuration.

If users find any part of the menu occluding the scene they are working on or would prefer the menu to be in a different position to suit their grip of the device, they may reposition it to any location on the screen; up to a third of the menu may appear off screen for better support of smaller devices. This is intended to improve the ergonomic use of HARATIO.

4.2.3 Visual Breadcrumb

One aspect of the radial design that disappears when implementing collapsible menus is the context of the user's path through the menu. As parent menus are collapsed to a state without menu items, it is unclear which items have been selected to reach the active child menu. Linear menu hierarchies used in WIMP (Windows, Icons, Menus, and Pointing device) interfaces preserve menu item selection via selection highlighting—traversing the menu three levels deep results in each parent menu indicating the item that lead to the subsequent child menu. Figure 4.8 illustrates an example from Microsoft Outlook, a common email client used in the workplace. The CRL menu adopts a similar approach by preserving the selection sequence of menu items through its hierarchy, which the authors note as a requirement of their design to support novice users in exploring the menu.

As discussed, the SAM, which provided inspiration for HARATIO's final menu structure, opts to display parent menus as rings surrounding the active menu. Menus in the SAM are divided into segments and, when selected, these are highlighted in yellow. The SAM preserves this highlight when displaying new child menus and so can provide a quasi-selection history by way of coloured ring segments. While the menu items themselves are not visible, the coloured segments maintain the positional consistency of the items within their respective menu.

HARATIO incorporates similar visual cues into the display of collapsed menu rings.

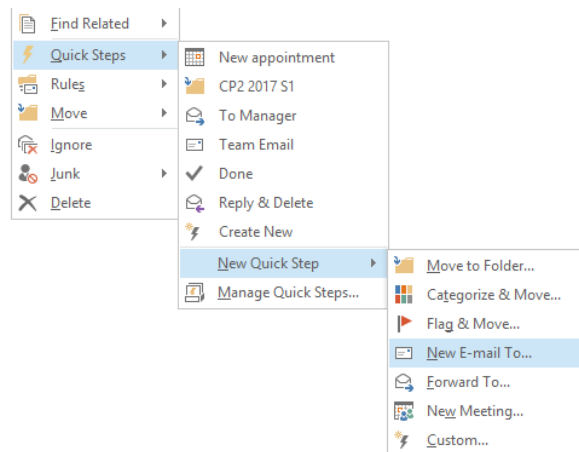


Figure 4.8: Linear hierarchical context menu displayed within Microsoft Outlook. In addition to child menu positioning, highlighting previous selections assists with indicating selection history.

Collectively, this is referred to as the *visual breadcrumb*. Just like the SAM, the relative segment of the selected menu item in each collapsed menu is highlighted—a lighter hue of the colour assigned to the menu is used—and indicates the user’s path through the menu. This occurs automatically as part of the menu activation animation discussed previously. The result is a breadcrumb trail of items selected that adds additional context in the form of a link between menus. The trail maintains the positional consistency of each menu item it represents by mapping to the items’ respective segments. An example of the visual breadcrumb is given in Figure 4.6b.

4.3 AR Mode

The default mode for HARATIO is *AR mode*. This is the mode the application initially starts in. AR mode displays a live feed from the device camera and includes active tracking. Users focus the device on various markers to view corresponding AR content, which is referred to collectively as a *scene*. If implemented by the author, scenes may be interacted with using the input modalities supported by the host device.

Figure 4.9 provides an overview of the AR mode interface. The interface scales automatically to accommodate the screen size of the device being used. In this mode, the radial menu is configured to show only the menu core and collapsed root menu. As described in section 4.2.2, the menu core houses a single button that comprises a pause icon and a label that instructs the user to select it to begin authoring. Selection of the button begins the transition process to author mode (described in section 4.4). If a marker has not been recognised, the menu core instead displays a visual indication that it is searching for one. Interaction with the menu in this state has no effect.

A status bar, displayed at the top of the screen, provides the user with information on the current state of HARATIO and the tracked marker. The left-hand side shows an indication of real time application performance (in frames per second) as well as the distance between the device and marker (in centimetres). If no marker is being tracked, the distance label is omitted. The right-hand side displays the current build version and a university identification for screen captures and demonstrations. The status bar is rendered

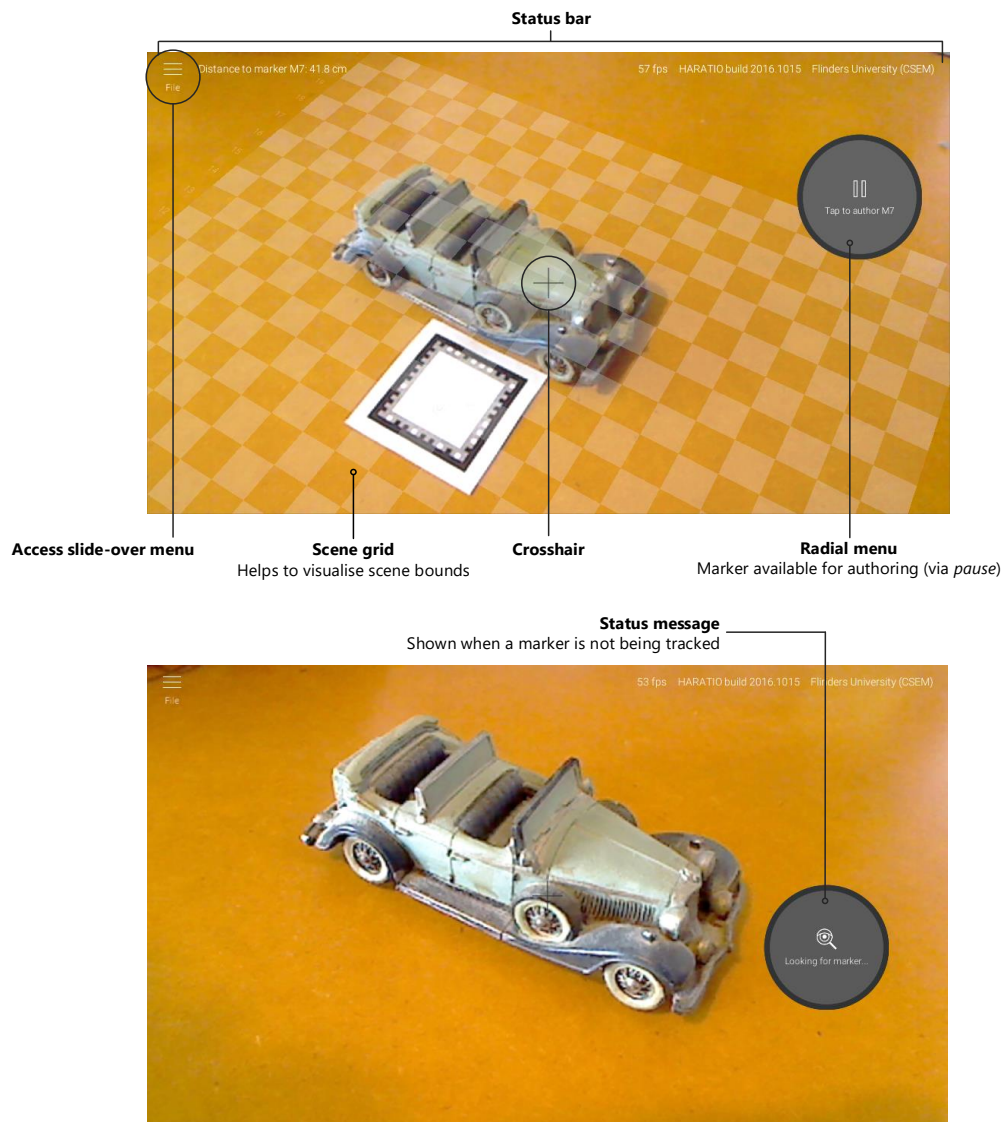


Figure 4.9: Overview of HARATIO's AR mode. The top image shows the interface when a marker is recognised. The bottom image shows the interface when a marker is not available.

on top of all other elements. The interface additionally provides a crosshair in the centre of the screen to assist users with positioning and aligning the device with respect to markers and scene content. The crosshair disappears automatically when transitioning to author mode.

The final element of the interface is the live, augmented view that comprises the device camera feed and registered virtual content (the scene). The camera feed is rendered to a background texture element that scales automatically to match the aspect ratio of the camera with that of the screen. This fills the entire display proportionally. The augmented view is projected below all other interface elements.

As with the MARS system (Höllner et al., 1999), HARATIO's AR mode interface incorporates the concept of screen- and world-stabilised elements. Though these terms have been previously discussed in relation to HMDs (see Feiner et al., 1993; Billinghurst et al., 1998)³, they still have relevance to handheld displays. Screen-stabilised elements are those that remain fixed in position regardless of changes to tracking data or device pose; within the AR mode interface, this corresponds to the status bar, radial menu, and the crosshair. Conversely, registered scene content is world-stabilised and changes position and orientation based on tracker data and device pose. Even when a marker without scene content is tracked, a world-stabilised grid is displayed (anchored to the marker) to assist the user with adjusting the position of the marker or device with respect to the bounds of the environment and desired scene area. The grid is discussed further in section 4.4.3.1.

Though screen- and world-stabilised elements are visible independently of one another, they maintain a relationship. The status bar displays the distance to the tracked marker, the radial menu enables authoring for the registered scene associated with the marker, and the crosshair may act upon the scene if used in a corresponding Behaviour script (discussed in detail throughout section 4.4.5).

4.4 Author Mode

The *author mode* provides a dedicated environment for novice users to create, edit, and script virtual objects within an AR scene. As discussed in the previous section, users enter author mode by first identifying target markers in AR mode and then initiating the transition process via the radial menu. Once the transition is complete, the authoring interface is revealed and the root menu activates to provide access to authoring options. The status bar, discussed in section 4.3, is preserved during the transition and remains visible above all interface components.

The authoring interface is divided into three main *activities* and scales to accommodate the device screen size as per AR mode. Each activity corresponds to a respective workflow that users are likely to perform while composing a scene. As well as providing a functional organisation to HARATIO's feature-set, the use of separate activities is intended to deliver a focussed experience in which only those items relevant to the current task are presented. Activities are accessed via corresponding buttons along the top of the root menu (Figure 4.6f) and result in the interface adapting to reveal any supporting components required. As discussed in section 4.2, users visually search menus top-down, so placing activity items in the top row minimises search time.

³ Feiner et al. referred to screen-stabilised as 'display-fixed' while Billinghurst et al. used the term 'head-stabilised'.

Authored AR scenes may be serialised to facilitate saving or loading from local device storage. Scene file operations are located within a slide over menu that is accessible using a File button visible in the top left corner of the author mode interface. The File button appears as part of the status bar and is accordingly visible on top of all other interface components. Discussion concerning scene serialisation and the slide over menu is provided in section 6.1.6 and 6.1.7, respectively, where their implementations relate specifically to the user evaluation conducted.

Throughout all authoring activities, users are provided with enhanced visual feedback in the form of notification messages that unobtrusively appear at the top of the screen. The messages provide explanation of application behaviour, updates concerning the status of scenes, or helpful hints regarding how activity operations can be performed. Saving an AR scene, for example, results in a notification being displayed that either informs the user the save operation was successful or provides explanation as to why it was unsuccessful. Notifications expire after a configured duration has elapsed or a certain condition has been met. No direct interaction is required. Expired notifications disappear in the same unobtrusive manner as they appear. HARATIO's notification system was motivated by feedback from user evaluations reported in Chapter 5, and further discussion on the implementation and design is provided in section 6.1.5.

The following describes the transition process between author mode and AR mode as well as specific interactions supported within the authoring environment. Interactions incorporate established touch gestures to support object selection and viewpoint manipulation, which are both necessary for efficient authoring.

4.4.1 Transitioning to Author Mode

The previous chapter discussed the occurrence of fatigue and frustration issues encountered by users while holding larger handheld devices in sub-optimal poses and simultaneously performing interactions. These issues were found to be exacerbated in cases where tracking content required an undesirable pose be maintained—for example, content located above the users' eye line. Similarly, smaller devices were observed to be susceptible to wobble caused by the counteracting force of a finger moving across the screen surface. These issues are mostly unique to the properties of handheld AR where the device provides both input and output functionality. While holding a handheld device with one hand affords interaction freedom via the ability to reach any part of the screen, it can also introduce instability to the interaction space that makes precise selection and manipulation difficult. Grasping a device with two hands helps alleviate this problem but greatly restricts the freedom of interaction to the thumbs and the areas of the screen they can reach. For larger devices, which almost necessitated being held with two hands, maintaining a secure grip while attempting to simultaneously interact with on-screen content can be challenging.

As discussed in Chapter 2, investigations focussed on improving the overall experience of handheld AR interaction frequently involve some form of pausing or *freezing* technique whereby all (Guven et al., 2006; Lee et al., 2009; Bai et al., 2012; Langlotz, Mooslechner, Zollmann, Degendorfer, Reitmayr and Schmalstieg, 2012; Vincent et al., 2013) or part (Arshad et al., 2016) of the scene is temporarily frozen in place to provide stability to the interaction space. In most cases, this is typically toggled manually by the user, but it

has also been explored implicitly (for small targets) as a factor of each interaction (see ‘Shift&Freeze’ by Vincent et al.).

Freezing the scene and preparing it for authoring comprise five steps that begin following the selection of the pause button. First, the camera feed is stopped. As the camera feed is rendered to a background texture element, stopping the camera suspends texture updates and preserves the last rendered camera frame. The next step involves caching the position of content associated with the marker in preparation for stopping the AR tracker. Every scene has a visual grid reference (discussed in section 4.4.3.1), so even if the scene contains no content, the registered grid position will be cached. Position caching works around the way Vuforia and Unity behave together when tracking is stopped and enables the position of scene content relative to the marker to be maintained. Without this step, the content would disappear. When caching has finished, the Vuforia tracker is stopped. Finally, the content is packaged as a writeable scene and made accessible to the three authoring activities. The interface also reconfigures itself with the root menu activating to reveal available authoring options. From the user’s perspective, the whole process is seamless: the frozen scene content remains registered with the frozen camera texture (using cached position data) in the same pose as when the transition process began. Authoring tasks may then be performed on the scene with the aid of view/scene stability.

Resuming to AR mode is accomplished via a complementary play button that is available within the radial menu core when both the author mode and root menu are active. Just as pushing the play button on a media player or remote control causes video playback to resume, selecting the play button on the radial menu results in the reverse of the aforementioned transition process being executed. When the AR tracker is restarted, all cached position data related to the frozen scene is discarded and control returns to the tracker—scene content effectively becomes read-only. Restarting the camera feed resumes updates to the background texture element. Finally, the interface reverts to AR mode and the root menu collapses to present the radial menu configuration described previously. Consequently, the play button is replaced by the pause button to indicate the scene is once again available for authoring. Figure 4.10 illustrates the transition process from AR to author mode.

4.4.2 Author Mode Interaction

Interaction within author mode can be categorised into two types: *object* and *viewpoint*. As the name suggests, object interaction relates to the selection and deselection of individual virtual objects. Selecting an object designates it the target for Edit and Behaviour activities and is achieved by tapping on it. Selected objects are displayed with a surrounding blue stroke. Deselecting an object occurs by either tapping it again; selecting another object, which designates *that* object the selected object; or tapping on an unoccupied area of the scene. Earlier iterations of HARATIO only permitted deselection using the first two options; however, this was later amended to include the third, as described in section 6.1.4, based on observations discussed in section 5.5.6. Currently, only one object may be selected at a time.

HARATIO adopts an ‘object-centric’ rather than ‘operation-centric’ approach when determining whether object selection changes should be allowed for any given authoring

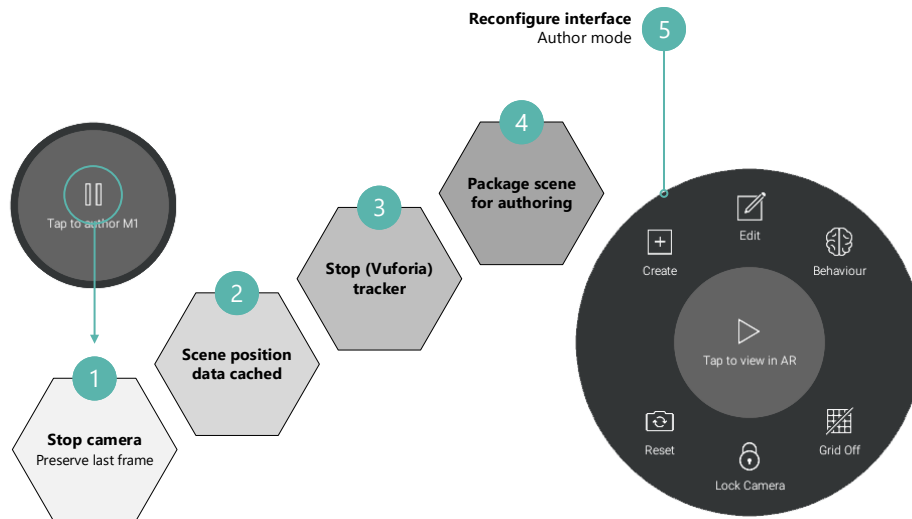


Figure 4.10: Illustration of steps involved in the transition from AR to author mode

mode configuration. In an object-centric approach, priority is given to the object. User first select the target object to modify and then access the menu corresponding to the desired activity operation to perform changes on it. By assigning priority to the object, it is clear where edits will be applied. Consequently, selection changes are temporarily disallowed while a modifying operation is in progress; that is, its menu is active. In contrast, an operation-centric approach would shift the priority to each individual operation such that users would first enter the associated menu and then select the target object. Although this approach allows selection to swiftly alternate between objects, there is more onus on the user to keep apprised of which object is currently selected to ensure edits are applied where they were intended—this also means being mindful of avoiding accidental screen contact that could lead to unwanted selection changes. In pursuit of simplicity and an approach suitable for novice users inexperienced with 3D environments, an object-centric approach was deemed preferable.

Objects located behind (occluded by) other objects can often be awkward to select. Doing so involves either temporarily moving the offending objects out of the way or transitioning back to AR mode, adjusting the device pose, and transitioning back to author mode. Neither are ideal options. One possible solution involves the use of additional interface components. [Wozniowski and Warne \(2011\)](#) present a menu to users when they perform selections to allow the desired object (occluded or not) to be chosen from a list. They also support the use of an additional fiducial marker as a world coordinate anchor to which objects are snapped, so rotating the physical marker can resolve the occlusion. [Mossel et al. \(2013b\)](#) proposed a two-step technique called DrillSample in which a list of objects, ordered according to original depth, is displayed via a separate interface whenever the user's selection results in the intersection of multiple overlapping objects.

HARATIO adopts a different approach that encompasses support for viewpoint interaction within author mode. Via a toggle control on the root menu (Figure 4.6g), users can unlock the authoring viewpoint and adjust it in four degrees of freedom (DOF) using a series of touch gestures. This enables instances of object occlusion preventing accurate

selection to be resolved by simply adjusting the point of view. Compared with DrillSample’s list-based approach involving separate interfaces, this method enables objects to continue to be selected in the context of their relative position within the scene (Figure 4.11). Supported viewpoint adjustments are illustrated in Figure 4.12 and include orbiting with a one-finger drag (b), panning with a two-finger drag (d), and zooming in (e) or out (f) based on the relative distance between two fingers.

Adjusting the viewpoint can quickly result in the appearance of visual conflicts (discussed in section 2.2) between the frozen scene content and the static background texture element. As illustrated in Figure 4.13, when viewing a scene in AR mode, the viewpoint and background texture remain fixed and it is the position and orientation of the scene that updates to reflect the tracked pose of the marker. In author mode, the scene is frozen in place using cached position data derived from the last tracked position; the background texture element is also frozen, retaining the last captured camera frame.

Certain viewpoint adjustments will result in static registration errors as the perspective of the scene no longer corresponds to the internal model of it. Rather than distort the background texture to match the adjusted viewpoint, thereby attempting to resolve the visual conflict, HARATIO instead reduces the texture’s opacity to 10% whenever a conflict will occur. Since viewpoint adjustments are anticipated to be infrequent and ephemeral, this approach was considered suitable as it maintains a visual connection to the physical environment and simultaneously declutters the view to assist with clarifying object selection. Opacity reductions are performed when any instance of an orbit adjustment occurs; panning or zooming are unaffected as the perspective between the background texture and scene content is offset but otherwise unchanged.

Quickly resetting the viewpoint to its default position and orientation can be achieved by performing a stationary one-section tap gesture anywhere over the scene (Figure 4.12c) or selecting a button on the root menu (Figure 4.6h). As resetting the viewpoint will remove any applied orbit adjustments, the background texture will return to opaque. Locking the camera will similarly reset the viewpoint as it restores the default configuration.

Earlier iterations of this functionality defaulted to an unlocked viewpoint when author mode was entered and concealed the background texture element in favour of an environment reminiscent of typical 3D modelling tools. This was amended to the approach described following the first user evaluation of HARATIO, and further discussion regarding the motivations for this are provided in section 6.1.1. The remainder of this section discusses the interface and feature-set provided by each of the three authoring activities.

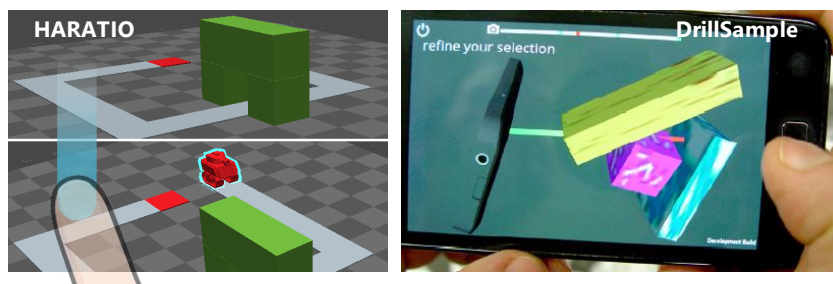


Figure 4.11: Two different approaches to resolving object selection occlusion. HARATIO (left) allows the authoring viewpoint to be unlocked and adjusted. DrillSample (right) (Mossel et al., 2013b) presents a list of affected objects in a separate interface.

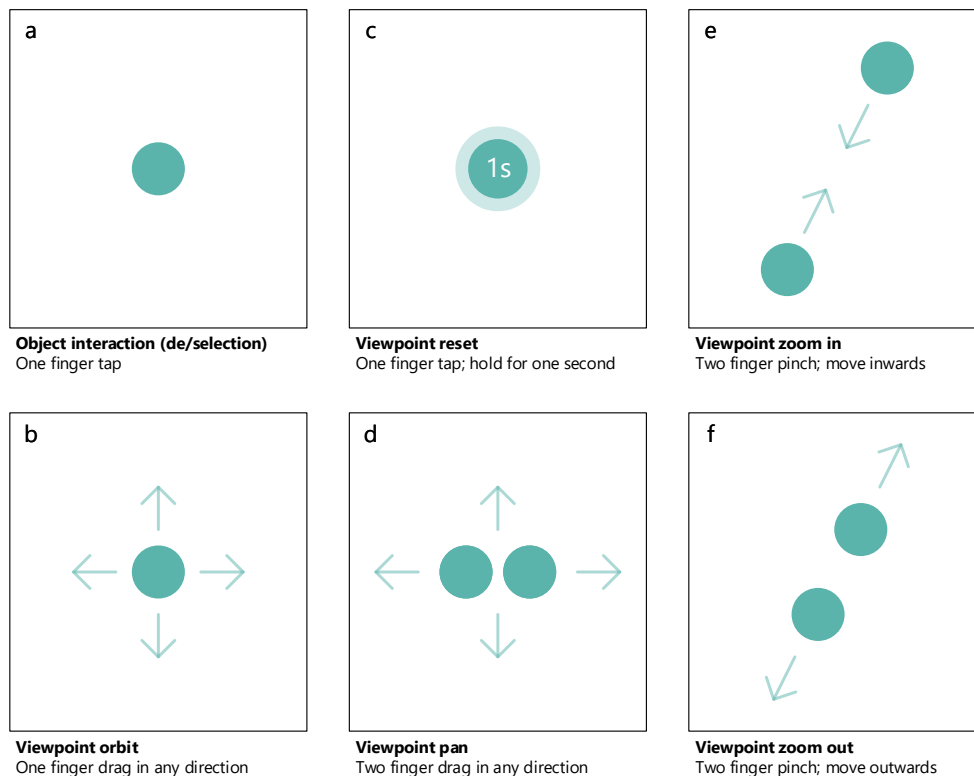


Figure 4.12: Supported touch interactions within author mode

4.4.3 Create Activity

The Create activity allows users to add one or more virtual objects to a scene from a fixed library. Basic geometric shapes and simple 3D models are supported. The library includes a cube, a sphere, a cylinder, a plane, and a tank (*Cartoon tank*, 2014). As shown in Figure 4.14, the collection of objects is presented as row of 3D icons in a panel at the bottom of the display. The icons provide a realistic representation of each object that is less abstract than a 2D icon.

As the Create activity includes no supplementary operations, the associated menu appears collapsed and displays only the assigned activity colour. The back button visible in the menu core can be used to exit the activity and return the user to the default authoring mode configuration and the root menu.

4.4.3.1 Scene Grid

A grid is included with every scene and provides a ground plane reference (matching the plane of the fiducial) to assist users with object placement and scene perspective. The grid is also a concession to the limitations of using fiducial markers as AR tracking aids. While fiducials are easy to deploy and well-suited to mobile platforms, they restrict the practical scale of augmentation. The device camera must be able to adequately capture and track the marker for associated content to be registered and displayed. Therefore, the size of the marker, and to some extent the quality and field of view (FOV) of the device camera, determines the trackable region—the ‘augmentable frustum’. The scene grid helps depict a sensible bounding area within this frustum and illustrates the possible locations where

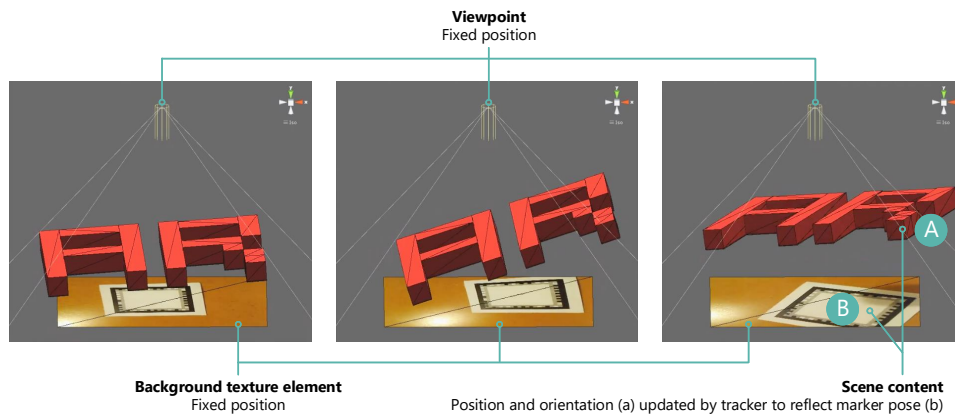


Figure 4.13: Relationship between background texture element, scene content, and viewpoint in AR mode

virtual objects can and cannot be located. It also safeguards users against placing objects outside the trackable region or otherwise on the fringe where continuous updates to pose data may affect their registration. With this in mind, the size of the grid used throughout development and testing has been determined based on the size of the markers and range of devices used.

Overall grid size is governed by cell size as well as the number of rows and columns. Each of these can be easily adjusted as necessary. Cell size effects the resolution of the scene and maps to the physical size of the marker used. A cell resolution of 1 equates to a single cell occupying the same area as the marker; a cell resolution of 0.5 equates to it occupying half, and so forth. The screenshots used throughout this chapter have been captured using a 5 cm² marker with a cell resolution of 0.3; therefore, 9 cells occupy the same area as the marker. Figure 4.15 illustrates the effects of cell resolution for the values described. The number of rows and columns can be set to determine the overall grid size based on the specified cell resolution. Odd row and column values can be tied with odd cell resolutions to ensure a cell always exists at the center of the scene. HARATIO uses a left-handed Cartesian coordinate system with x and y axes running parallel to the grid plane and the z axis orthogonal. This is consistent with the Unity coordinate system.

To provide users with a reference to individual cells within the grid, the final rendering is adorned with labels denoting each row and column number. When designing a scene, users inexperienced with working in a 3D environment may find it beneficial to first sketch out their design on graph paper before translating it to the grid within HARATIO. The row-column reference provided by the corresponding labels is intended to assist with this.

The scene grid is visible by default whenever HARATIO is in author mode. Visibility may be toggled via an option on the root menu to quickly allow users to preview their scene (Figure 4.6a). When in AR mode, the grid is only displayed if the tracked marker has no scene content associated with it (i. e. it is a *new* scene). In this case, the grid is designed to suggest a blank canvas onto which content may be authored. If the user is placing the marker themselves within the environment, the visibility of the grid also aids in locating the marker with respect to the work area and surrounding physical objects.

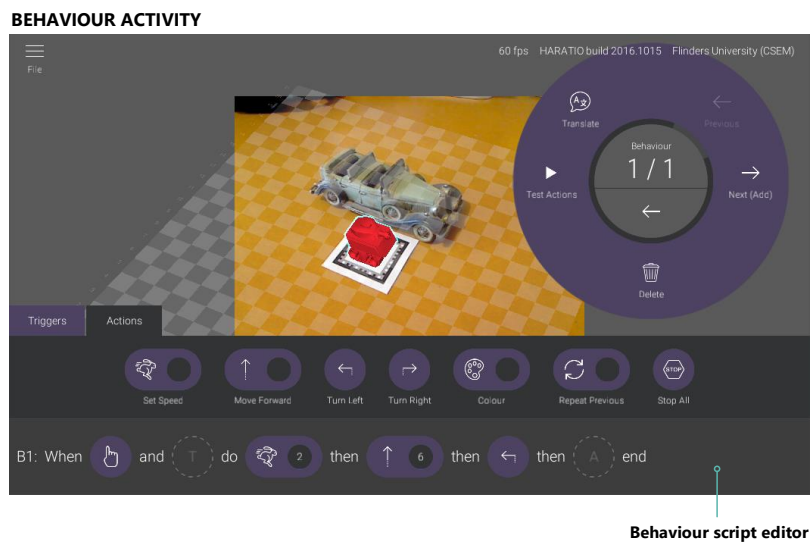
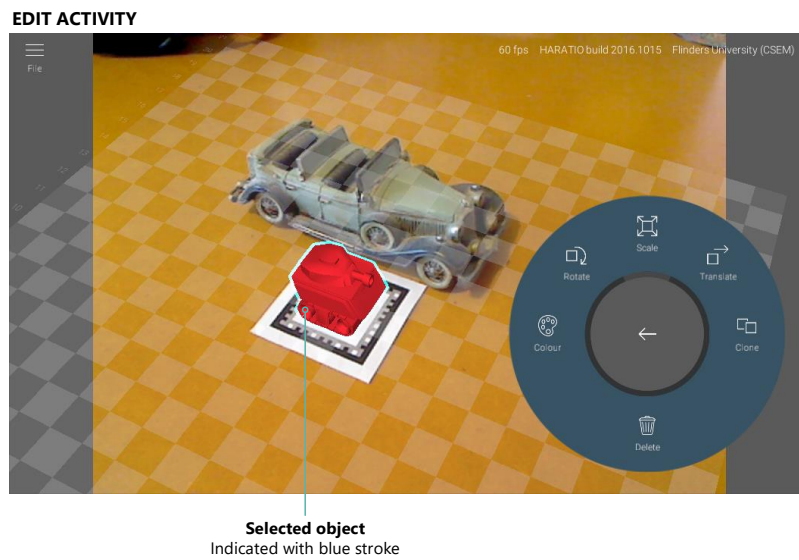
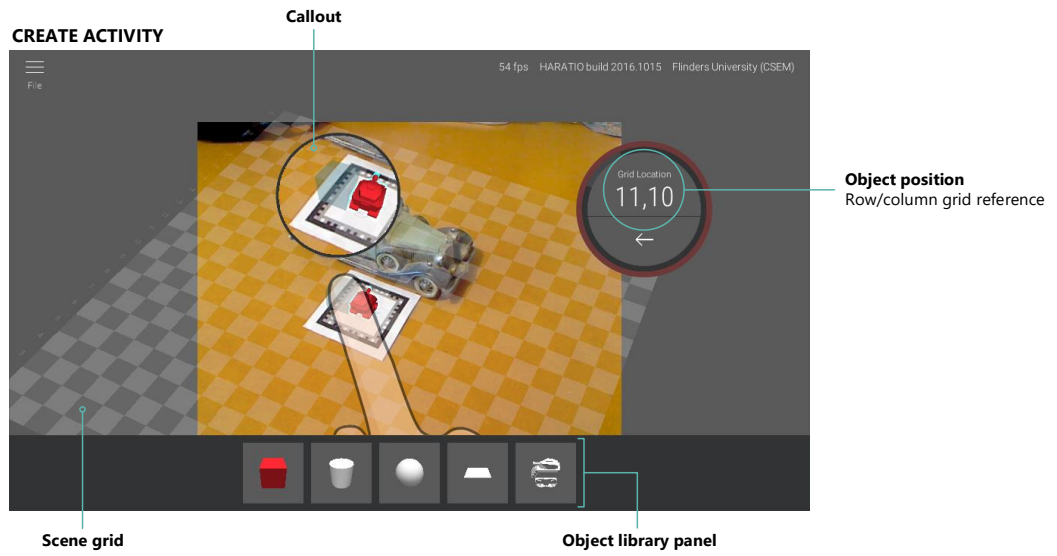


Figure 4.14: Overview of author mode interfaces for *create*, *edit*, and *behaviour* activities.

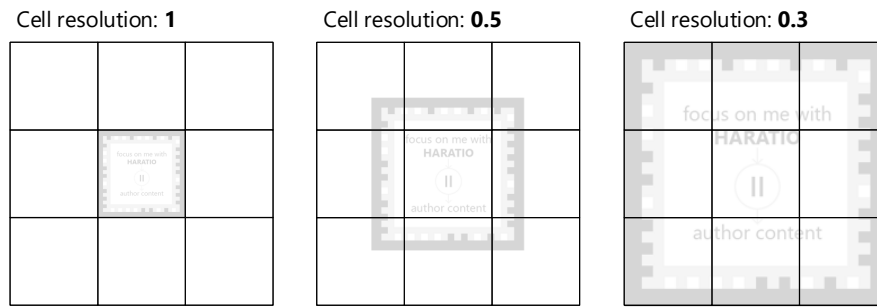


Figure 4.15: Mapping of scene grid cells to physical fiducial marker size

4.4.3.2 Object Creation

Objects added to a scene must be located on a grid cell. This can be accomplished in one of two ways, denoted as:

- O1. Tapping the object's icon in the object library panel
- O2. Dragging from the object's icon to a destination grid cell

If the user performs O1, an instance of the selected object will be created immediately on the centre grid cell. If they instead perform O2, a 'ghost' of the selected object will be anchored to the finger point (relative to the grid plane) as it is dragged along the screen surface to a target grid cell. The ghost object will snap to the nearest grid cell automatically to aid with positioning and help mitigate the relative low accuracy of touch input. If an object is to be located on a grid cell already occupied by an existing object, the new object will be positioned on top of it (in reference to the z axis) to prevent ambiguities that would result from two objects occupying the same space.

It is inevitable when performing a direct touch interaction that the finger, hand, or some part of the wrist will, to some extent, obscure part of the screen. This becomes especially noticeable when interactions are of undefined duration. When a new object is created, it is initially scaled to the equivalent of one grid cell—the minimum size for an object. As the ghost object used in O2 is a facsimile of the final object, it also shares this scale. Even with the aid of cell snapping, the finger performing the interaction may partially or fully obscure the view of the object being manipulated along with possible target grid cells. HARATIO implements a technique based on 'Shift' (Vogel and Baudisch, 2007) to provide the visual assistance necessary to overcome this issue. Whenever an object is added to the scene using O2, a callout is rendered above the finger point to reveal the occluded content beneath. The callout contains a copy of the screen area masked by the finger, magnified by a factor of approximately two, and remains fixed above the finger while it moves across the screen surface. As the object being placed is anchored to the screen point controlled by the finger, the callout serves to clarify its location within the scene and the grid cell it might occupy. The grid cell the object is currently above is also displayed centrally within the radial menu core as a row-column grid reference that, coupled with the use of cell snapping, avoids the need to show a selection point within the callout. Completing object placement and dismissing the callout both occur when the finger lifts off the screen, similar to the take-off strategy (Potter et al., 1988). Figure 4.14 shows the callout in action with O2; without clarifying the screen area beneath the finger inside the callout, the position of

the ghost object, a cube, would be ambiguous with respect to its placement adjacent to the tank object.

The use of a callout to assist with object placement allows interaction during the creation process to remain direct. An otherwise offset ghost object would break this relationship by requiring a mapping between touch point and screen point.

4.4.4 Edit Activity

The Edit activity provides users with the means to perform actions that either act on the (selected) object or one of its properties. The default activity interface is provided in Figure 4.14. Each of the available editing actions is listed on the Edit activity menu (Figure 4.6j). In line with the item ordering discussed in section 4.2 for the root menu, the most common editing operations are placed at the top of the menu (*Rotate*, *Scale*, and *Translate*). Some operations require further configuration or user input. In these cases, additional child menu are displayed accordingly to accommodate such options.

4.4.4.1 Colour

HARATIO includes seven predefined colour swatches for object customisation. An eighth *Occlusion* swatch is also available and is discussed below. Swatches are presented within a dedicated child menu (Figure 4.6i) that is accessed via the *Colour* button located on the Edit activity menu. Selecting a colour swatch will immediately assign it to the selected object. As the colour menu is auto-collapsing, this action will also cause it to collapse and return to the Edit activity menu without the user needing to manually select the back button in the core. This facilitates rapid colour changes. Should the user wish to exit the colour operation without changing the object's colour, the back button can be selected to return to the edit menu manually.

Occlusion colour In addition to correct registration, one of the challenges of providing a convincing AR experience is implementing correct occlusion between virtual and physical objects such that the two appear to coexist. The eighth colour available on the colour menu, *Occlusion*, exists to provide better support for this situation. When the final scene is composited using the device camera feed, occluding physical objects with virtual objects is straightforward. Such virtual objects can be rendered at a higher depth order than the background texture element giving the appearance that they are positioned *in front of* physical objects. The reverse is more problematic. A typical camera image does not contain depth information that can be used to selectively omit parts of virtual objects that are intended to be occluded by physical ones. Some form of physical environment modelling is therefore required so the system can correctly interpret the depth of objects within it.

Approaches to calculating depth information and resolving occlusion have included generating depth maps of the environment (Wloka and Anderson, 1995; Breen et al., 1996), evaluating contours within an image to determine whether they are in front of or behind virtual objects (Berger, 1997), registering suitable proxy models (phantoms) in place of physical objects (Fuhrmann et al., 1999), and identifying and refining occlusion boundaries using edge-based tracking data (Klein and Drummond, 2004). More recently, depth sensing cameras have also been explored. Leal-Meléndrez et al. (2013) used the



Figure 4.16: Example occlusion between physical and virtual objects: DART (left) (MacIntyre et al., 2004) and HARATIO (right).

Microsoft Kinect sensor to retrieve real time depth information and subsequently remove occluded parts of virtual objects. While the Kinect is not particularly practical for mobile use, handheld devices conforming to Google’s [Project Tango](#) specification include depth sensing cameras built-in, enabling them to perform similar depth estimation in real time. The Lenovo Phab 2 Pro and Asus ZenPhone AR (both smartphones) are examples of two devices that implement this technology.

Of all the different approaches, the simplest is to use virtual objects as proxies for physical objects. The virtual proxy objects function as replicas of their physical counterparts and are positioned and sized accordingly within a scene. As the proxies serve only to provide depth information, they may be omitted from final scene composition; that is, rendered to the z buffer only. This approach has limitations, however, and can be subject to visual artefacts if the proxies do not closely match their physical counterparts or tracking anomalies affect their registration.

A technique incorporating proxies was implemented in the Designer’s Augmented Reality Toolkit (DART) (MacIntyre et al., 2004). DART denotes object actors as ‘virtual’ or ‘physical’, which refer to either normal 3D objects or objects intended to model real world items, respectively. Physical actors are rendered to the z buffer only. As the left image in Figure 4.16 illustrates, the real world coffee mug appears to correctly occlude the virtual flower pot, which is achieved using a physical object actor that is aligned with the mug. In this case, a cylinder is used and serves as the proxy. When the final scene is composited, the result is the flower pot appears to be partially obscured by the mug.

The occlusion colour available within HARATIO implements the proxy method using a dedicated material that can be assigned to any virtual object. The material functions as a depth mask by incorporating a custom graphics shader that effectively renders whatever it is assigned to invisible by discarding all colour channel data (RGBA). As the shader writes to the z buffer and is positioned earlier in the render queue than standard colours, anything located behind that intersects the current perspective will be culled. If the material is assigned to a proxy object representing a foreground real world object, any virtual objects located behind will appear correctly occluded. The background texture element captured from the device camera remains unaffected by use of the occlusion colour (depth mask) as it is processed earlier in the render queue than all virtual objects. An illustration of the render queue ordering used by HARATIO with respect to the predefined Unity render queues (Unity Technologies SF, 2014b) is given in Figure 4.17. HARATIO’s render queues operate in parallel to Unity queues. Render tasks with the same queue ID—indicated to

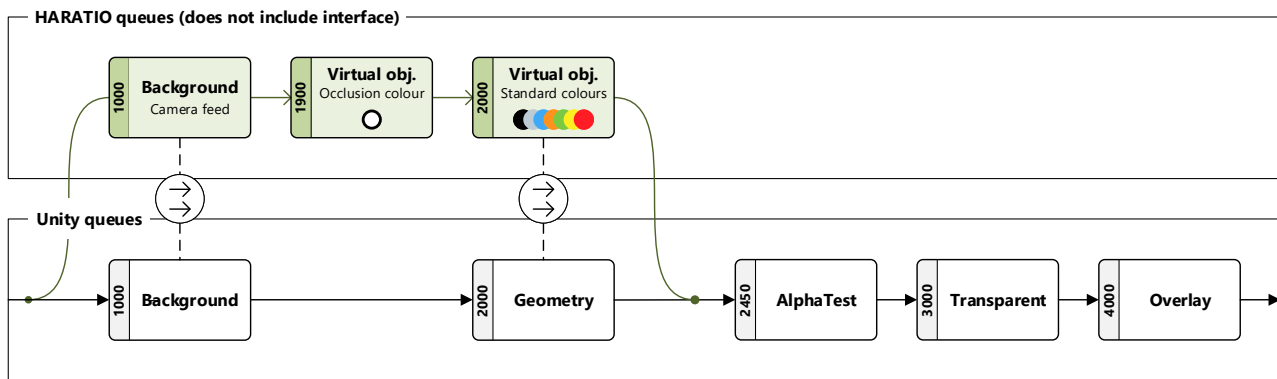


Figure 4.17: HARATIO render queue ordering in reference to predefined Unity render queues. Queue IDs are displayed to the left of each render task. Lower queue IDs are processed by the GPU first.

the left of each render task—execute concurrently.

The DART example described previously can be replicated within HARATIO using the occlusion colour and is shown in the right image of Figure 4.16. In this example, the flower pot has been replaced with a tank model and the mug with an ornament, but the physical object is similarly represented using a proxy virtual object to provide the missing depth cue. This was achieved by creating a cylinder object to serve as the proxy and positioning it where the physical ornament was located relative to the tank model. The cylinder was then assigned the occlusion colour to ensure it functioned as a depth mask when the final scene was composited.

A key advantage of this approach is that it operates alongside existing authoring methods and does not require the user learn a separate, dedicated workflow. Virtual object proxies can be created for their real world counterparts and then assigned the occlusion colour so they can be used for depth information without appearing in the final scene composition. From the user’s perspective, proxy objects are created just like any other object and the occlusion colour is assigned just like any other colour. Other than a basic understanding of the functional difference between occlusion and standard colours, users do not need to learn new skills to incorporate object occlusion into their scenes. A useful consequence of using proxies in this way is the added reference they provide for the occluding physical object, which can be used to aid with the placement of surrounding virtual content.

Objects assigned the occlusion colour are displayed with a semi-transparent preview material within author mode to distinguish them from standard coloured objects and make them easily identifiable. The semi-transparent nature of the preview material provides an indication that the object will serve as a proxy in the final scene and will not be rendered as per standard colours. Selection of a proxy object will turn the preview material opaque to provide a clear indication of selection free from other objects partially visible behind. Initial implementations of the occlusion colour did not include a separate preview material and instead rendered proxy objects in author mode using the described depth mask method. This made it difficult to visualise where proxy objects were located unless they were selected or revealed by partially masked surrounding objects. Based on feedback received from the user evaluation of HARATIO reported in Chapter 5, the occlusion colour implementation was modified to incorporate the author mode preview material as described. This is further

discussed in section 6.1.2.

4.4.4.2 Rotate, Scale, and Translate

Modern touch-driven 2D interfaces often incorporate a series of direct gestures to support object rotate, scale, and translate (RST) operations. A common example of their use can be found in many photo applications where the user is able to use one or two fingers to directly manipulate (Shneiderman, 1983) an image. Moving two fingers, most often the thumb and index finger, clockwise or counter-clockwise around an arbitrary centre point performs a rotation. Moving those fingers towards or away from one another—commonly referred to as a ‘pinch’ gesture—performs a scale. And moving a single finger along a vector performs a translation. These gestures provide an affordance that mimics physical interaction with real world objects. They work well for 2D environments because the screen surface is also 2D. The finger contact points are therefore able to remain tightly coupled to the object. Wellner (1993) demonstrated an early implementation of similar gestures (in a 2D AR context) with the DigitalDesk system, which combined physical and digital elements on a work surface. Users could select parts of physical documents by defining a selection rectangle with two fingers (akin to a scale gesture) and then copy the corresponding content to a digital document using a single finger dragging motion.

Although intuitive and widely understood, attempting to leverage similar RST gestures within a 3D environment, such as the one HARATIO provides for authoring, can be problematic. Given the extra DOF, it is often difficult for multi-finger contact points to be *inside* the target object and the gestures themselves become more ambiguous. For example, while rotation of a 2D object can only occur around *one* axis, a 3D object can be rotated around any one of *three* axes. The same is true for axial translation.

Various methods to support manipulation of 3D objects for touch screens and mobile devices have adapted existing RST approaches or incorporated supplementary input mechanics. Reisman et al. (2009) extended the principles of traditional RST gestures for use in 3D environments by supporting other DOF with additional contact points. In the case of rotation, this involved using two fingers from one hand to define the rotation axis and a single finger from the other to rotate the object around it. HOMER-S and 3DTouch (Mossel et al., 2013a) utilise the physical position and orientation of a handheld device to directly control the manipulation in six DOF or define the target axis for touch gestures. These two techniques have been specifically designed for use in single-handed AR scenarios. Microsoft’s 3D Builder (Microsoft Corporation, 2014) adopts a similar approach to typical CAD and 3D modelling tools where a gimbal widget is presented around the object in separate RST states. Users perform RST operations by directly manipulating the corresponding gimbal axes with either mouse or touch input. The ‘tBox’ (Cohé et al., 2011) extends the idea of a gimbal widget by displaying a wireframe cube around objects that users directly manipulate with touch gestures to perform RST operations. The tBox supports nine DOF.

Given HARATIO’s target audience, a key concern with providing 3D object manipulation support is simplicity. Where possible, RST operations need to be presented with an affordance and discoverability that complex gestures and intricate device movements are unlikely to provide. Displaying a gimbal widget around the object assists with affor-

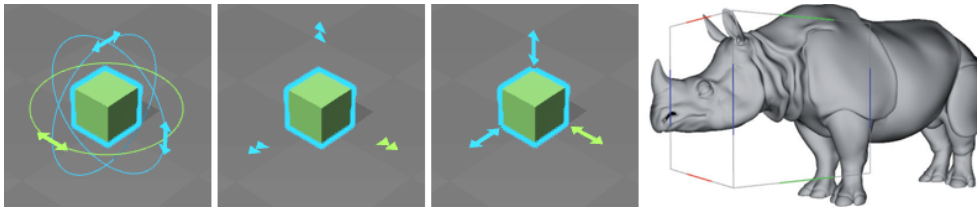


Figure 4.18: Microsoft 3D Builder (Microsoft Corporation, 2014) gimbal widgets in rotate, scale, and translate states. The manipulation of each widget arrow corresponds to movement along the associated RST axis (the active axis is displayed in green). The rightmost image shows the tBox widget (Cohé et al., 2011).

dances by indicating the possible motions of the object in each RST state; however, directly performing these with multi-finger gestures on handheld devices is susceptible to screen occlusion and the general inaccuracies of touch input. As discussed in section 4.4.3.2, solutions such as Shift are appropriate for overcoming screen occlusion with single finger interactions but would present challenges in multi-finger scenarios as the placement of the callout would be complicated by the motion of the gesture and the constraints of a small screen. While the tBox widget implements measures to counteract these issues⁴, and has been reported to be usable by both expert and novice 3D modellers alike, it has arguably lower affordance than traditional gimbals due to the inability of the wireframe cube to deliver the same degree of contextual visual cues. Figure 4.18 contrasts the differences.

In pursuit of a simplistic solution, HARATIO instead provides separate child menus for each RST operation that employ both direct and indirect interaction. The menus govern how the operations are performed and additionally include supplementary feedback in the form of object information displayed in the menu core. The rotation operation, for example, displays the object's angle in reference to the orientation of the scene marker.

Translation is kept direct as it doesn't make sense to support indirectly: menu controls would be too cumbersome to use and lack the intuitive command over movement that exists with a direct approach. Using the plane of the scene grid as a guide, translation is supported in two DOF and is performed by the same one-finger dragging gesture that users use to initially place objects in the scene. As objects must be placed in the scene before they can be edited, users are pre-exposed to the interaction required for translating them. Incorporating direct interaction into the translate operation presents a conflict between *object* and *viewpoint* manipulation. Translation uses the same gesture as viewpoint orbiting (Figure 4.12b), so there is an implicit ambiguity regarding whether the interaction should affect the object or the viewpoint. To disambiguate this situation, changes to the viewpoint orbit are temporality disabled while translation is active. As the user has purposefully selected an object and entered the translate menu, an intent to manipulate the object not the viewpoint has been expressed. Viewpoint pan and zoom gestures remain unaffected, and orbit is re-enabled as soon as the translation operation is deactivated.

Instead of using direct interaction, rotation and scaling are both achieved indirectly via custom controls within in each respective child menu. Knoedel and Hachet (2011) investigated the differences between direct and indirect RST actions on touch screens and found the two techniques perform comparably in 3D environments. While they concluded

⁴ One example is supporting translation of an object via selection of any cube edge running parallel to the target axis. The user can select the edge that offers the least object occlusion or is positioned most ergonomically.

direct interaction was still faster overall, indirect interaction was discovered to provide higher levels of efficiency and precision. Given HARATIO's target audience and associated design goals, an indirect solution can offer the following advantages:

1. The discoverability of the operation is improved and is further supported by an interface that users see as soon as they activate the associated menu.
2. Building on (1), better affordance can be provided via custom controls for the operation. As discussed later in this section, the control handles are fixed to a track that suggests how they may be manipulated. Moving the handle along the track rotates or scales the object.
3. Screen occlusion is mitigated. Users aren't required to use multiple fingers to interact with the object, which would otherwise increase the risk of obscuring parts of it. If the menu obscures the object, it can be moved to a different screen location. A callout is difficult to use with multi-finger gestures as the ideal placement of it would be subject to constant variation.
4. Conflicts with viewpoint manipulation are avoided. Viewpoint manipulation can be maintained where interactions outside the menu control the viewpoint and interactions within the menu control the object.

As the display of gimbals offers good awareness as to possible RST manipulations, HARATIO displays a separate passive gimbal around the target object for each RST operation, similar to those shown in Figure 4.18. The passive gimbals provide a visual connection to the associated radial menu state, and changes to RST options will update them accordingly. The remainder of this subsection discusses the interface and interaction for rotate, scale, and translate operations.

Rotate As discussed, rotation occurs indirectly. Users activate the rotation operation by selecting the `Rotate` button on the Edit activity menu (Figure 4.6j). The radial menu then presents a new child menu that displays a custom slider control for performing rotation operations on the selected object.

As shown in Figure 4.19, the slider control is comprised of two constituent parts: a circular *track* and a *handle* that is visually affixed to it. Movement of the handle around the track maps to the rotation angle of the object. With reference to the orientation of the scene marker, the twelve o'clock position represents an object rotation of $0/360^\circ$, the three o'clock position a rotation of 90° , and so forth. The current angle of the object is displayed centrally within the radial menu core in 1° increments, and a passive rotation gimbal is displayed around the object to indicate the rotation axis and the space the object will occupy at various angles.

While manipulating the handle, the track colour brightens to provide visual feedback to the user that the mapping between handle position and object rotation is in effect. As HARATIO's authoring system is built on the alignment of objects to a grid, rotation angles are currently restricted to increments of 90° . The user may move the handle to any arbitrary angle, and the object will update accordingly, but releasing the handle (lifting the finger off the screen) in between a multiple of 90° will cause the rotation to snap to

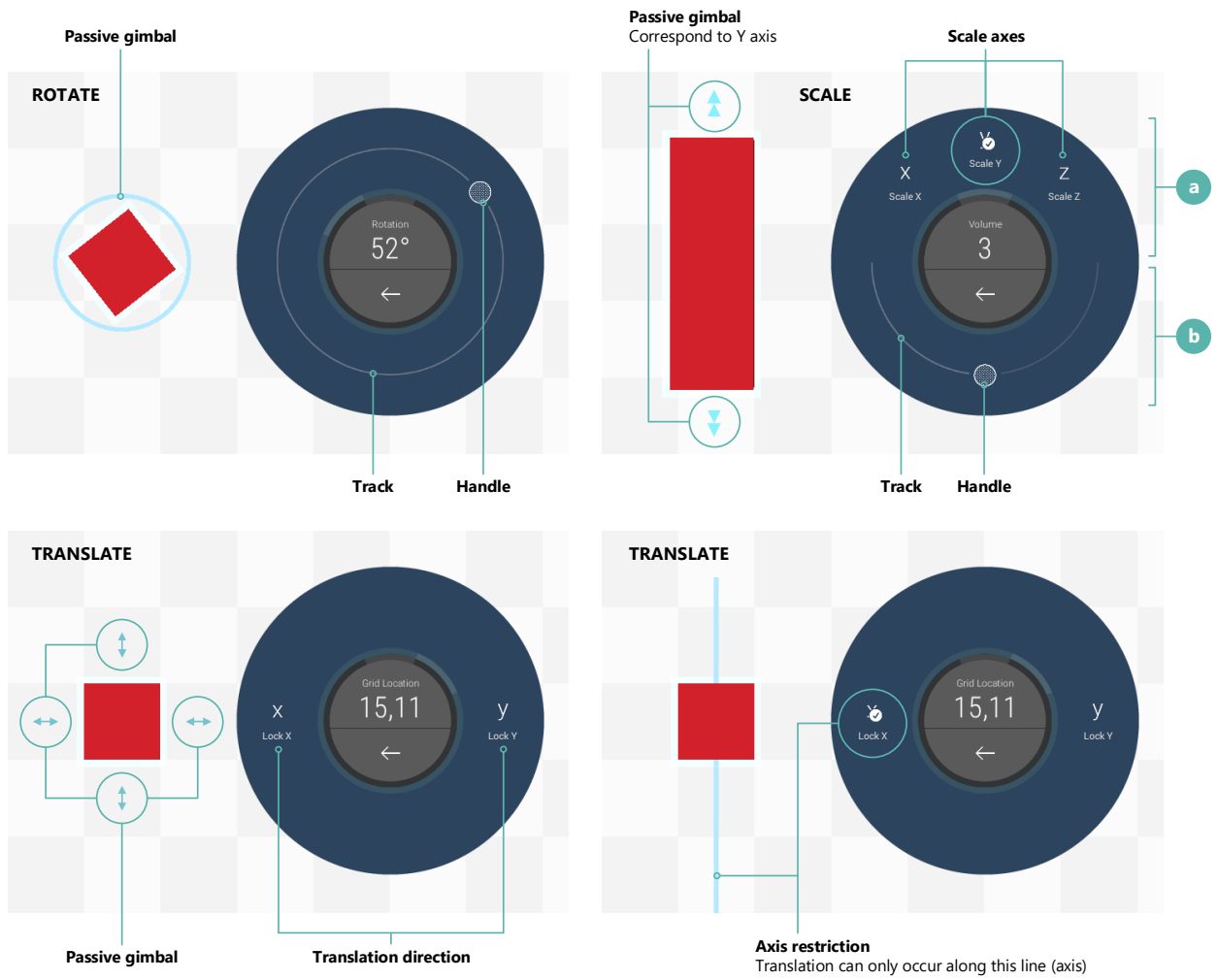


Figure 4.19: Illustration of rotate, scale, and translate menus

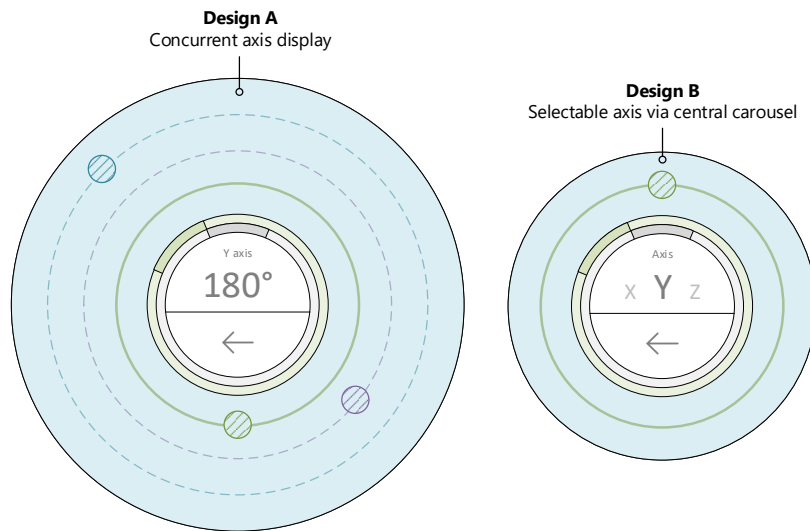


Figure 4.20: Two indirect rotation menu designs supporting three axis rotation. Design A: all three axes are displayed concurrently by separate tracks. Design B: the central area of the radial menu displays a carousel that users swipe through to select the active axis. The single track/handle controls rotation around this axis.

the nearest increment. Thus, if the handle is moved to 30° and released, the object will snap to a rotation of $0/360^\circ$; if the handle is moved to 45° , the object will snap to 90° .

In the prototype versions of HARATIO evaluated, rotation was only supported around the z axis (R_z); that is, the axis orthogonal to the scene grid. As users would be building scenes from the ground up using the grid as an alignment and construction aid, rotation around other axes was not necessary. Although not implemented, potential designs for indirect three-axis rotation were considered and are illustrated in Figure 4.20. Two alternate designs are shown. Each leverages the same interaction mechanic of a slider control comprising a handle and track. The first design (a) expands the menu size to include additional slider controls for the remaining two axes. In this case, each control would be associated with a separate axis and assigned a unique colour to distinguish it. The colour would also be reflected on the passive gimbal displayed around the object. Moving one of the handles would cause the radial menu core to update its content to reflect the angle for the corresponding axis. While design (a) allows quick access to all axes, the additional screen space it occupies would likely be unsuitable for use on smaller devices, and the presence of three slider controls appears cumbersome. The second design (b) maintains the original size of the rotation menu and instead replaces the upper portion of the radial menu core with a carousel of axes. Users would swipe through each axis, one at a time, to select the desired one to manipulate. The track colour would update according to the selected axis as per design (a). Moving the handle would similarly cause the radial menu core to update its display to denote the current angle, temporarily replacing the axis selection carousel.

Scale As with rotation, scaling is also performed indirectly using a slider control available within the scale menu, which is accessed by selecting the **Scale** button on the Edit activity menu. Objects are scaled uniformly along S_x , S_y , and S_z axes. These may be toggled independently to provide the user with control over how scaling will be applied.

The scale menu is composed of two sections, denoted as (a) and (b) in Figure 4.19. Section (a) contains three toggle controls (section 4.2.2) that represent each of the three scale axes. The object will only be scaled along axes corresponding to active toggle controls. A passive scale gimbal displayed over the object updates in accordance with the toggle controls to further clarify scale directions. In the example in Figure 4.19, only the S_y axis is active and the gimbal therefore displays arrows in this direction only.

Section (b) contains a slider control resembling the one used in the rotate menu. However, instead of the track providing a full 360° of rotation, it presents a 180° arc with the handle positioned at its centre. The handle is used to adjust object scale based on its movement direction. Moving the handle to the right increases object scale while moving it to the left decreases object scale. The centre position represents a ‘dead zone’ in which no scale is performed and extends $\pm 10^\circ$ to prevent accidental operation. The handle is biased to the centre of the track (its starting point) and will spring back to this point when not being manipulated. Scaling occurs uniformly at a rate of one grid cell per second. At every scale increment, the object’s size will increase or decrease by half a cell in each axis direction. To prevent objects scaled in the S_z direction appearing partially below the scene grid, such operations are accompanied by a simultaneous (silent) translation operation equal to half the scale value in the positive T_z direction.

Objects may be scaled to a maximum size that does not exceed the scene grid in any dimension and a minimum size no smaller than one grid cell. The radial menu core displays the current object size as a volume in grid cells cubed (gc^3) and updates dynamically as scale is adjusted.

Translate Translation is also activated by selecting the **Translate** button on the Edit activity menu. The translation menu is shown in Figure 4.19. As translation occurs directly, there are no specialised controls that uses manipulate to perform the operation. Instead, the translation menu displays two toggle controls that enable users to restrict the translation direction in order to assist with object positioning. A passive gimbal is displayed over the object with arrows indicating the direction it may be moved, which, by default, is any direction along the scene grid.

As discussed, users perform a translation directly by dragging the selected object with their finger to the desired location on the scene grid. Objects are snapped to the nearest grid cell, and the radial menu core dynamically updates to display the current grid location in row-column notation. If the object size is unmodified from its default value (one gc^3), a callout is additionally shown above the finger point (discussed in section 4.4.3.2) to overcome screen occlusion issues and assist with positioning. Objects larger than the default scale are less susceptible to being completely obscured beneath the user’s finger.

Like scaling, axial translation can also be constrained. The menu provides two toggle controls that support restricting the translation direction to T_x or T_y . When the translation direction is restricted, the passive gimbal is replaced with a line that passes through the object and indicates the direction in which it can be moved (Figure 4.19). Tang et al. (2015) incorporated similar visual axis extensions into their authoring tool to help with object placement. Translating an object with an active constraint will cause it to move only in the corresponding direction regardless of any movement in other directions. This replicates the functionality of directly manipulating gimbal axes discussed earlier where

the object moves only along the axis being controlled. Translation restrictions are mutually exclusive. If the direction is restricted to T_x when the T_y restriction is activated, the T_x restriction will be automatically deactivated.

Vertical translation (what would be considered T_z) is currently unsupported. When considering evaluations of HARATIO, it was envisaged users would sculpt scenes from the ground (grid) up by creating and positioning base objects and then building upon them to construct compound models or features—like the way a house is constructed by first laying the foundation. It was therefore not expected, nor desired, that users would need to position objects in ‘mid-air’. As such, translated objects are automatically positioned on top of any existing objects that already occupy the same target grid cell. This prevents objects being lost inside other objects.

4.4.4.3 Additional Operations: Clone and Delete

Two additional operations are provided on the Edit activity menu to facilitate management of scene objects in a holistic manner—these operations effect the object as a whole rather than individual properties.

The first of these is object cloning, which enables users to quickly add copies of the selected object to the scene. Cloned objects retain the physical properties of their source: type, size, rotation, and colour. Position is also copied with the distinction that the cloned object will be located on top of the source object, rather than inside it. Behaviour scripts, which are discussed in detail in section 4.4.5, can be thought of as describing an object’s personality. They are considered unique to each object and are therefore not copied. This is analogous to the idea that cloning an animal, for example, would result in an identical physical copy but independent behavioural traits and personality. Cloning is performed by selecting the Clone button available on the Edit activity menu (Figure 4.6j). Once complete, the newly cloned object is automatically selected.

While cloning enables copies of the selected object to be quickly added to the scene, the delete operation removes the selected object from the scene. Deleting an object is accomplished by selecting the Delete button at the bottom of the Edit activity menu. The button is purposely separated from non-destructive editing functions to prevent accidental selection. Deleting an object is final and will cause the Edit activity menu to auto-collapse—Edit activity operations are no longer able to act on an object once it has been deleted—returning the user to the root menu where they can then choose to create new objects or edit others in the scene.

4.4.5 Behaviour Activity

In addition to providing the tools necessary to create and edit virtual scene objects, HARATIO also supports per-object interaction via *behaviour scripts*. Used previously in APRIL (Ledermann and Schmalstieg, 2005) and DART to describe dynamic elements, the term *behaviour* was selected to promote the idea of adding intelligence and personality to otherwise static objects within an AR scene. As shown in Figure 4.14, the Behaviour activity interface includes a dedicated editor panel that is used to create and manage scripts for the selected object. The process of authoring scripts is based on the appropriate arrangement of various graphical symbols using a drag-and-drop interaction mechanic.

Each symbol represents a *trigger* or *action* clause that responds to an event or performs an action. Symbols added to a script are surrounded by assistive labels that collectively form the behaviour construct and are intended to aid with readability and comprehensibility compared with typical programming languages. The menu for the Behaviour activity, shown in Figure 4.6k, includes options to cycle through different scripts, preview them in action, or translate them to complete natural language sentences. Translation of behaviour scripts occurs in real time and provides a description of what the script will do.

4.4.5.1 Behaviour Language

As discussed in Chapter 2, adding interactivity to AR scenes has been explored to various degrees in authoring tools such as DART, ComposAR (Seichter et al., 2008), and MARAT (Rumiński and Walczak, 2013). DART provides a collection of Lingo⁵ scripts authors can add to a scene—or extend if desired—but they exist within the bounds of the host PC application. In the case of ComposAR, scripting involves using Python to program object interactions, which requires knowledge of the language’s syntax and grammar; some degree of previous programming experience is therefore a useful, if not necessary, skill.

As MARAT targets mobile devices and users without such programming skills, it instead aims to completely remove the need to engage with a coding environment by providing a very high level of abstraction. A user progresses through lists of predefined functions to configure interaction properties for objects, and these may be further parametrised to suit their intended use. While this approach offers simplicity in terms of defining interactivity without the need to write code, it sacrifices flexibility: the user can only select from a single function on each list and is therefore unable to combine functions to build advanced interactions. Furthermore, as the lists are presented separately, they are unable to be viewed together in context.

When considering scripting capabilities for HARATIO, which also targets novice users without programming skills, a point somewhere between ComposAR and MARAT was desired. A code oriented scripting language was considered too complex and arguably too impractical for handheld use. Lists of predefined functions are inflexible and restrict the scope of rules that can be created. Visual programming languages offer a potential middle ground by providing a level of abstraction that retains much of the flexibility of textual languages while offering a less intimidating appearance. Examples such as Scratch (Maloney et al., 2010), Alice (Cooper et al., 2003), Hopscotch (Hopscotch Technologies, 2017), Lego Mindstorms (The LEGO Group, 2017), and Flip (Howland and Good, 2015) typically present programming structures as distinct blocks that users arrange accordingly to describe the program they wish to create. This form of language presentation greatly reduces the need to learn abstract keywords, syntax, or language-specific semantics. It also provides an implicit level of type and logical error safety by ensuring blocks can only be connected and parameterised in valid ways.

HARATIO’s behaviour script language is inspired by Kodu (MacLaurin, 2011), a visual programming language developed by Microsoft Research and motivated by many of the examples mentioned previously. Like Scratch, Kodu is designed around teaching young children introductory programming concepts through independent exploration

⁵ Lingo is a scripting language used by Macromedia Director (now Adobe Director).



Figure 4.21: Two Kodu rules captured within Kodu Game Studio (Microsoft Corporation (Research), 2017). Rule 1 is complete and describes changing an object's colour to orange when left-clicked with a mouse. Rule 2 is empty.

and self-directed learning. While other visual languages offer more advanced capabilities and closely replicate core programming structures, Kodu aims to be simpler, focussing instead on concise expression of ideas. It is a highly abstract, event driven language that draws inspiration from robotics and intelligent agents (Stolee and Fristoe, 2011). Kodu is used within Kodu Game Lab (Microsoft Corporation (Research), 2017), a dedicated programming environment that adopts a game-development-within-a-game approach. Users build a 3D game world and then define rules for objects within it using a visual editor. The rules may be evaluated (debugged) by playing the game and then further refined as needed. In 2014, Microsoft incorporated the Kodu language into Project Spark, an evolution of Kodu Game Lab with more mature capabilities. The product increased the exposure of the language to users of Xbox consoles in addition to PCs.

Kodu uses the concept of 'sensors' and 'actuators' to define rules that govern how objects behave. Sensors in Kodu relate to what objects can respond to or detect. They may be refined by filters and selectors to narrow their effect. Actuators represent actions that are performed in response to sensor activation. They may also be refined using modifiers to specify parameterisation. A complete rule is comprised of a sensor clause, made up of a sensor and one or more filters/selectors, and an actuator clause, made up of an action and one or more modifiers. The Kodu language is a context-free language (CFL) and can be expressed as a context-free grammar (CFG) (see Stolee, 2010).

Each component of a Kodu rule is represented as a visual card, and users arrange these in meaningful ways to express the type of rule they wish to create. Kodu's interface provides a template for new rules with dedicated areas for sensor and actuator clauses. Cards can be added to each clause via a context sensitive pie menu that displays options relevant to the current state of the rule. Each rule can only contain one sensor and one actuator clause, so the first time the menu is invoked for the sensor clause, it provides a list of available sensors. The next time, it provides a list of filters, and so on. Figure 4.21 shows two rules in the Kodu editor. The first represents a complete rule with valid sensor and actuator clauses; the second represents a new, template rule.

For the design goals of HARATIO, Kodu's simple sensor-actuator rule approach was considered well-suited. The visual cards and absence of typing translate well to the modalities of touch interaction and the various handheld screen sizes HARATIO supports. Furthermore, the abstraction afforded by using cohesive visual cards, in conjunction with a rule appearance that shares similarities to conditional expressions, promotes the

formulation of interactions in a manner akin to the way instructions are given. Consider a foreign traveller asking a local resident for directions to a nearby tourist attraction. Casting aside any language barriers, providing directions would generally take the form of several step-by-step actions that should be followed in order. For example:

1. Begin by heading down road *x*
2. When you reach street *y*, turn right; and so forth

This analogy maps well to the use of Kodu style rules as each instruction can be considered a distinct expression and further separated into constituent sensor and actuator components. Although novice users may lack programming skills, the concept of injecting virtual objects with various instructions in this fashion is likely to be relatable, intuitive, and less intimidating.

MacLaurin (2011) describes the similarity in operation of sensor and actuator clauses as being analogous to the conditional *if...else* statements used in procedural programming languages. To improve affordance and readability, *when* and *do* labels are prepended to the presentation of each clause within the editor. As each card is also displayed with an accompanying label, the when-do labels can help with interpreting the context of the cards assigned to the sensor and actuator clauses, respectively. The use of clear iconography and descriptive tooltips also help. Without these, rules are open to ambiguity. For example, the sensor clause text *when mouse left* from the first rule in Figure 4.21 doesn't directly indicate that it is the left mouse button that must be clicked—it just says *left*. It could just as easily be misinterpreted as *when the mouse is moved left*. In Kodu Game Studio, the use of accompanying icons and tooltips helps disambiguate such cases. When considering the behaviour system for HARATIO, this was also a point of consideration: how can users clear up ambiguity in the event it arises? Labels often need to be kept short for space reasons. Relying on icons alone is unlikely to be sufficient as comprehension of iconography can vary between users of different age groups (Koutsourelakis and Chorianopoulos, 2010) or ethnicity (Auer and Dick, 2007). And tooltips are obviously impractical when the input mechanics rely on touch interaction. HARATIO instead adopts a translation feature for scripts that serves to further aid users and can be toggled on or off as needed. This feature is discussed in section 4.4.5.5.

HARATIO's behaviour script syntax maintains the base format of sensor-actuator clauses and when-do labels. Sensors in the behaviour grammar are referred to as *triggers* while actuators are referred to as *actions*. These terms were considered more descriptive of each component's intent and less confusing given the target audience. Kodu sensors and actuators were themselves sometimes referred to as 'conditions' and 'actions' (Stolee and Fristoe, 2011) while other tools discussed in Chapter 2 adopted comparable terminology: 'cues' and 'actions' in DART and 'sensors', 'triggers', and 'actions' in ComposAR. Using this syntax, a basic behaviour script can be expressed as⁶:



The when-do labels are treated with the same precedence as triggers and actions; they appear inline and help enforce a structure for the behaviour that mimics a simple sentence

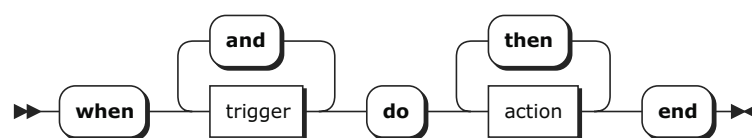
⁶ Syntax diagrams created using Railroad Diagram Generator (Rademacher, 2017)

phrase. The word *end* is appended to the behaviour to clearly signify when the script will terminate; that is, following the final action. This is functionally similar to the way semicolons and braces are used in C-based languages to denote the end of a statement or control block. While a punctuation mark like a full stop would have more closely adhered to a grammatically correct sentence structure, it was considered something that would be too easily overlooked. In this instance, a word such as *end* was considered better placed to unequivocally signal the conclusion of a script.

Each behaviour must contain at least one trigger and one action. As with Kodu, triggers are conditions that must be satisfied before any actions can be performed. Trigger conditions may be tied to physical hardware (sensors), AR tracker state, user interaction, or internal HARATIO state. A tracker-based trigger, for example, could be configured to only activate once the device has been positioned within a certain distance of a marker. Actions occur in response to triggers and represent something that is done or performed. They can act on the state of the object, the execution of the parent behaviour script they belong to, or other behaviour scripts. Example actions include changing the host object's colour or stopping all currently running behaviour scripts attached to it.

Both triggers and actions are declared as either *basic* or *configurable* types. The difference between the two relates to their respective support for parameterisation. Like Kodu's use of filters and modifiers on sensors and actuators, configurable types make use of user defined parameters to control operation. A timer trigger intended to delay action execution would require a parameter to specify the delay time and would therefore be implemented as a configurable trigger. A trigger intended to detect object selection, which is a binary condition, would operate without parameterisation and would therefore be implemented as a basic trigger.

Combinations of triggers and actions may be defined within a behaviour script to enable advanced functionality. This is a point of difference with the Kodu language where only a single sensor and actuator clause may be added per rule, and the concept of filters, selectors, and modifiers are used to narrow their effect. When a behaviour contains multiple triggers, all must be satisfied before any actions are performed. Multiple triggers are separated within the behaviour script by the word *and* to communicate this relationship and retain the readability of the script's sentence derived structure. Conversely, actions occur sequentially, so a behaviour containing three actions would execute action one, then action two, and finally action three. The exception to this rule is sequence modifying actions that may alter execution order. An action that performs a repeat operation would be one example as it would cause actions earlier in the chain to be performed again. Multiple actions are separated within the behaviour script by the word *then*, again, to communicate the relationship between them and retain readability. This separation also aids in ensuring triggers and actions maintain reasonable spacing and hit target sizes appropriate for touch interaction. The dispersion of *and* and *then* labels throughout the script happens dynamically as multiple triggers or actions are added. A behaviour script containing a series of triggers and actions would therefore be expressed as:



Algorithm 1 Trigger check routine

```
1: coroutine CHECKTRIGGERS( $B$ )
2:   loop
3:      $T_{satisfied} \leftarrow 0$ 
4:     for all  $T \in B.Triggers$  do
5:       if  $T.Evaluate()$  then                                      $\triangleright$  Returns true when trigger is satisfied
6:          $T_{satisfied} \leftarrow T_{satisfied} + 1$ 
7:         continue                                                $\triangleright$  Check next trigger
8:       end if
9:       if  $T.CanInvalidate()$  then
10:        for all  $T2 \in B.Triggers \neq T$  do
11:           $T2.Invalidate()$                                         $\triangleright$  Invalidate all other triggers
12:        end for
13:        break                                                    $\triangleright$  Don't check further triggers following invalidation
14:      end if
15:    end for
16:    if  $T_{satisfied} = count(B.Triggers)$  then
17:       $B.IsTriggered \leftarrow true$ 
18:      yield break                                                $\triangleright$  End coroutine
19:    end if
20:    yield return WAITFORSECONDS( $s$ )                              $\triangleright$  Pause for  $s$  seconds then resume
21:  end loop
22: end coroutine
```

When a behaviour contains multiple triggers, the state of one trigger may affect other triggers in the behaviour and cause them to become *invalidated*. As all triggers in a behaviour script must be concurrently satisfied (valid) before any actions can be performed, invalidation helps preserve the meaning expressed by the adjoining *and* labels and the overall intent of the script. Each trigger has an internal flag that determines whether it can invalidate other triggers. When a trigger that can invalidate is determined to be unsatisfied, all other triggers within the same behaviour will be notified regardless of where they are positioned in the ordinal sequence. If they respond to invalidation messages, they will reset. Evaluation of any remaining triggers will then cease. When the behaviour is next processed, evaluation will restart from the first trigger in the sequence. It is not until all triggers are concurrently satisfied that the behaviour script's actions will begin executing. A description of this process appears in algorithm 1 where B refers to the behaviour script being processed, T and $T2$ the triggers belonging to the script, $T_{satisfied}$ the number of triggers that have been satisfied, and s the delay in seconds between trigger evaluations.

As an example to help clarify this concept, consider a cube object that contains a single behaviour script with two triggers, *focus* and *wait*. The behaviour is intended to change the cube's colour to blue after the host device has been focused on it for five seconds. The user begins by positioning the device over the object, causing the *focus* trigger to be satisfied and the *wait* trigger to begin its countdown. The user then prematurely moves the device away from the object before the *wait* trigger countdown has finished. At this point, the *focus* trigger condition would become unsatisfied; however, without some form of invalidation, the *wait* trigger countdown would continue elapsing until it reached zero and its state changed to satisfied. If the user later moves the device back over the object, the behaviour's actions would immediately begin executing as both trigger conditions would be considered satisfied even though this was not the original intent of the behaviour. Invalidation allows the state change of one trigger to affect the validity of others. The *focus* trigger is one such trigger that can invalidate. Thus, when the *focus* trigger becomes

unsatisfied, the wait trigger will receive a notification and, as it responds to such messages, reset its countdown. With this in mind, the focus trigger would need to remain satisfied for the duration of the wait trigger's countdown so as not to reset it. This preserves the intent of the behaviour and ensures its actions are only performed at the correct time.

Figure 4.22 shows the CFG for behaviour scripts and includes all possible triggers and actions (currently implemented) in the language. The CFG, expressed in Backus-Naur Form (BNF), serves to delineate the individual behaviour components and describe how they fit together to form a complete and valid script. Variables (non-terminals) are listed on the left-hand side of productions while available triggers and actions (terminals) appear on the right. Behaviours ultimately start with an AR scene that contains the necessary virtual objects for behaviour scripts to be attached to. A scene may contain multiple objects and a single object may contain multiple behaviour scripts. Each behaviour script is comprised of the when-do-end labels and appropriate combinations of triggers and actions. Each trigger and action may be further classified as either basic or configurable.

Each object within a scene is limited to five behaviour scripts, and each script may include up to three triggers and twelve actions. These limitations were imposed keep the creation of behaviours manageable for the user evaluations discussed in Chapters 5 and 6. The limits are arbitrary and can be adjusted as necessary.

4.4.5.2 Triggers

Table 4.1 summarises the list of triggers available in the current language. While not exhaustive, the selection represents a range of triggers necessary for completing the tasks performed during user evaluations of HARATIO discussed in the following chapters. The table describes each trigger's type, whether it can invalidate other triggers, and whether it responds to invalidation messages. The icon used to represent each trigger within the interface is also included⁷. Additional triggers may be added as required using HARATIO's application programming interface (API). Further details on each trigger are provided in Appendix B.1.

4.4.5.3 Actions

Table 4.2 summarises the list of actions available in the current language. As with the previous listing of triggers, the selection represents a range of actions necessary for completion of evaluation tasks. The table describes each action's type as well as whether it may modify execution order (sequence). The icon used to represent each action within the interface is also included. Additional actions may be added as required using HARATIO's API. Further details on each action are provided in Appendix B.2.

4.4.5.4 Behaviour Editor

The process of creating a behaviour script is based on a drag-and-drop philosophy in which triggers and actions are meaningfully arranged to create the desired outcome. The bulk

⁷ As with the radial menu iconography (section 4.2.2), icons denoting individual triggers and actions were sourced from Icons8 LLC (2017).

⟨scene⟩ = ⟨object⟩ | ⟨object⟩ ⟨scene⟩
 ⟨object⟩ = ⟨behaviour⟩ | ⟨behaviour⟩ ⟨object⟩
 ⟨behaviour⟩ = when ⟨trigger⟩ do ⟨action⟩ end
 ⟨trigger⟩ = ⟨trigger-basic⟩ | ⟨trigger-config⟩ | ⟨trigger⟩ and ⟨trigger⟩
 ⟨trigger-basic⟩ = Always | Tap | Focus
 ⟨trigger-config⟩ = Wait | Distance
 ⟨action⟩ = ⟨action-basic⟩ | ⟨action-config⟩ | ⟨action⟩ then ⟨action⟩
 ⟨action-basic⟩ = Stop All | Turn Left | Turn Right
 ⟨action-config⟩ = Move Forward | Set Speed | Repeat | Colour

Figure 4.22: CFG of HARATIO’s behaviour language. Expressed in Backus-Naur Form.

Table 4.1: Summary of available triggers. In the context shown, *Tx* denotes the ability of the trigger to send invalidation requests and *Rx* denotes the ability to receive them.













NAME	TYPE	TX INVALIDATE	RX INVALIDATE	DEPENDENCY	ICON
Always	Basic	No	No	-	
Wait	Configurable	No	Yes	-	
Distance	Configurable	Yes	No	AR tracker	
Tap	Basic	No	Yes	Touch input	
Focus	Basic	Yes	No	AR tracker	

Table 4.2: Summary of available actions

NAME	TYPE	SEQUENCE MODIFYING	ICON
Move Forward	Configurable	No	
Turn Left	Basic	No	
Turn Right	Basic	No	
Set Speed	Configurable	No	
Colour	Configurable	No	
Repeat	Configurable	Yes	
Stop All	Basic	Yes	

of the creation process takes place in the editor that appears at the bottom of the screen whenever the Behaviour activity is active.

The editor is separated into two primary sections. The top section (Figure 4.23a) consists of a tabbed shelf that houses a horizontal list of available triggers and actions. Each trigger and action is represented by a graphical symbol, which consists of an icon contained within a circular capsule. A label below the symbol indicates its name. Symbols may appear in one of three sizes depending on whether the associated trigger/action (TA) is basic or configurable. Basic TAs appear as a simple circle. Configurable TAs require user configuration before being used, so they instead appear as elongated circles with an input field displayed to the right of their icon. The input field is initially blank to suggest that something—the configured value—is missing. When the TA has been configured, the input field displays an appropriate value indicating the active configuration. The nature of the value determines the size of the field and whether the symbol appears as per Figure 4.23 (b) or (c). The two configurable symbols shown in Figure 4.23 illustrate examples of possible values: the duration, in seconds, for a `Wait` trigger and the distance, in centimetres, for a `Distance` trigger. This is intended to provide feedback to the user on the current state of the TA without them having to manually enter the configuration panel.

The bottom section of the editor (Figure 4.23d) consists of the working area used to build scripts. Scripts are displayed in their component parts with the textual labels discussed in section 4.4.5.1 (when-do-end) interspersed between placed TA symbols. When adding a new behaviour script to an object, a template is provided to outline the basic structure of the script. The template includes placeholder symbols in the locations where TA symbol placement is valid. Initially, this is after the when and do labels. Placeholder symbols appear as a dotted outline of a basic TA symbol and contain either a T or A designation to indicate whether they accept a trigger or action.

Users build scripts by dragging desired TA symbols from the tabbed shelf onto appropriate placeholder symbols within the working area. When the user initiates a dragging operation on a symbol, its colour changes from purple to grey to indicate that it has been picked up; its label is also hidden. When the symbol is positioned over a valid placeholder with a matching type, the colour will return to purple and the placeholder will disappear to indicate a valid drop location. If the symbol is located over an invalid placeholder, the colour remains grey and the placeholder does not disappear. As well as providing a hint as to where TA symbols can be placed, the use of placeholder symbols ensures a basic level of type-safety and prevents the occurrence of syntax errors. Consequently, an action symbol cannot be inadvertently placed where a trigger symbol should go, and vice versa. Once a TA symbol has been successfully located on a valid placeholder and the user has completed the drop action, the working area will refresh to display the updated script with the symbol now part of the behaviour. If the respective TA limits have not been reached, a new placeholder will then become available after an adjoining *then* or *and* label. The process is repeated until the script is complete. If the script overflows past the available horizontal screen space, the user may scroll the working area using a drag gesture. Scroll position is maintained between script refreshes.

If the TA symbol added to the script is configurable, a modal configuration panel will automatically appear on top of the shelf to facilitate configuration. By appearing on top of the shelf, the user is encouraged to complete the configuration before moving on to

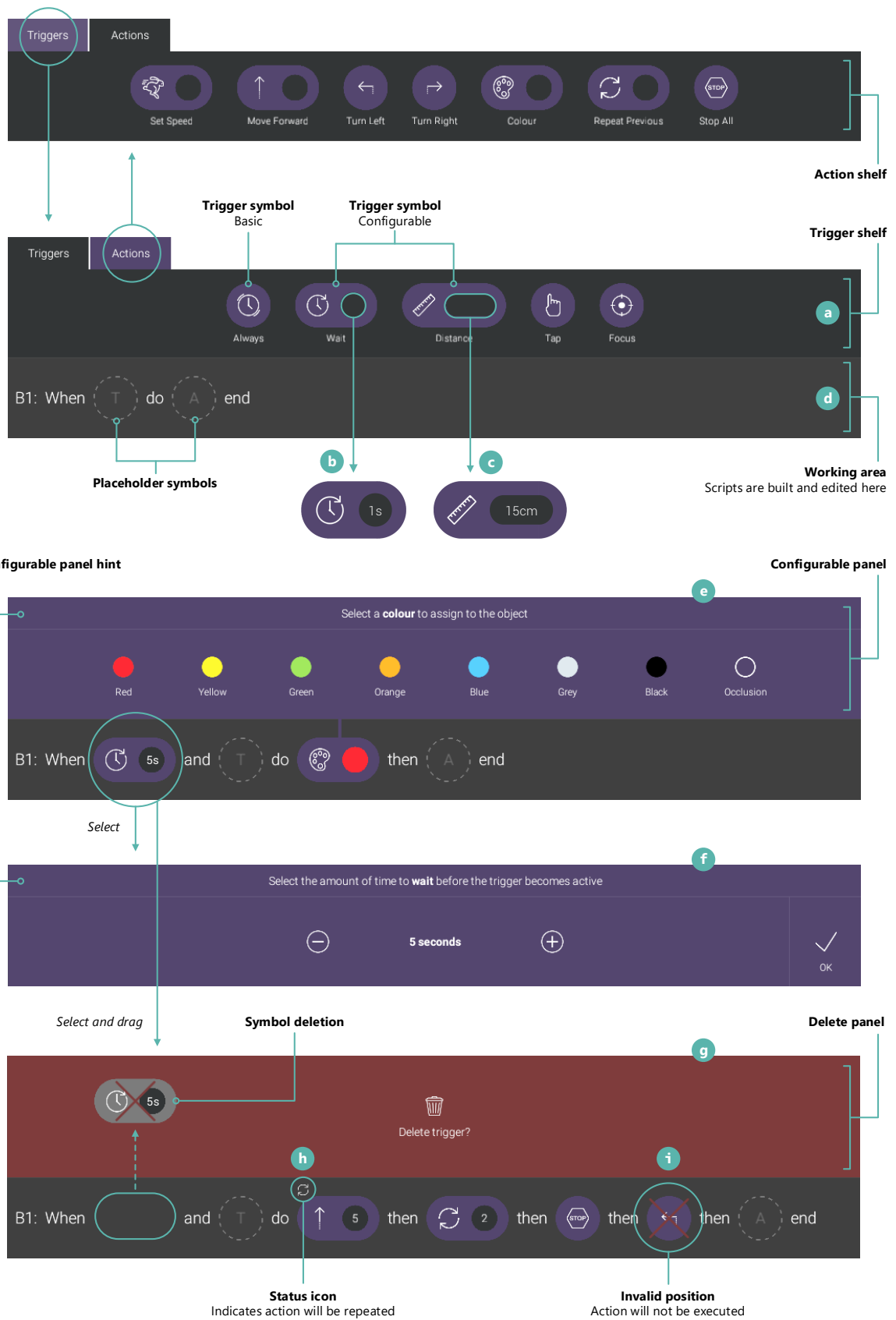


Figure 4.23: Overview of behaviour script editor interface. The editor comprises a series of panels that appear as necessary to facilitate script creation and modification.

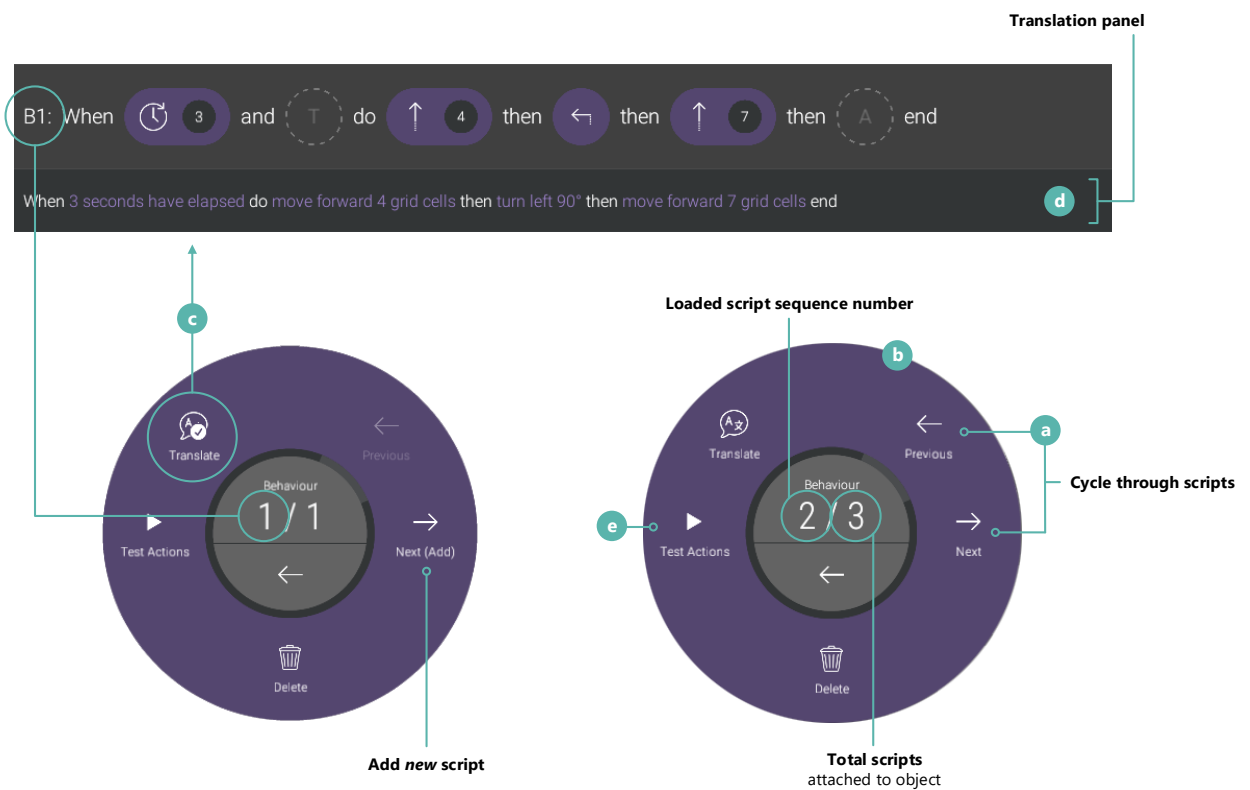


Figure 4.24: Overview of translation panel and Behaviour activity menu

add additional TA symbols. As configurable TAs remain invalid until properly configured, this is an important step. The panel is linked to its host symbol by a line that clarifies the relationship within the working area.

All configuration panels include a hint message that instructs users on how to configure the TA and what the selected value represents. The top example (Figure 4.23e) shows a configuration panel for the `Colour` action; the hint message instructs users to select the colour that should be assigned to the object when the action executes. The colours appear in the same order as displayed clockwise on the colour menu (section 4.4.4.1). Selection of a colour dismisses the panel and completes the configuration. The colour symbol shown in the working area then refreshes to display the selected (configured) switch as the background of the input field. If the user later wishes to change the configured colour, the configuration panel may be redisplayed by tapping the associated symbol within the working area.

All other configurable TAs are numeric-based. In this case, the configuration panel displays the current value—or a default value is if one has not been set—surrounded by two buttons that increment or decrement it (Figure 4.23f). Numeric-based TAs internally specify a lower and upper bound along with a step value to ensure only valid within-range options are selected. Unlike the colour configuration panel, users confirm the selected value and dismiss numeric panels by selecting the `OK` button anchored to the right edge. As before, completing the configuration causes the selected value to appear within the input field of the associated symbol in the working area. The value unit is abbreviated to

ensure the value can be adequately shown.

Symbols added to a script may be reordered or removed if the user wishes to modify the execution sequence or correct a mistake. Changing the position of an existing TA symbol within a script is accomplished via a similar drag-and-drop interaction to the way they are added. When relocating a symbol, the user drags it to a new proposed position, and the working area dynamically creates the space necessary to house it if the position is valid. Once the symbol is located over a valid new position, it may be dropped to apply the reordering. This will cause the working area to refresh to display the change. Symbols are only able to be repositioned within the same group of matching TA types, so it is not possible to relocate an action symbol to a position reserved for triggers, and vice versa. The same applies for placeholder symbols, which accept new TAs only, not existing ones.

Deleting a TA symbol begins in the same manner as reordering. When beginning a reorder operation, a delete panel will appear on top of the shelf much in the same way a configuration panel appears when a new configurable TA is inserted. If the user wishes to remove the TA from the behaviour instead of relocating it to a different position within the script, they can do so using this panel. The panel displays a hint message that instructs users on how to use it. The message is accompanied by an icon of a rubbish bin, which is used consistently throughout HARATIO's interface to denote destructive actions (see section 4.4.4.3). Once a symbol is positioned within the bounds of the panel, a red 'X' overlay appears on the symbol, the panel background changes to red, and the hint text updates to present a confirmation message (Figure 4.23g). Completing the operation by dropping the symbol inside the panel deletes it from the script. The panel will then disappear and the working area will refresh to display the updated script.

As discussed in section 4.4.5.1, the processing of actions is sequential except for a handful of sequence modifying actions. As their name suggests, sequence modifying actions can alter the action processing chain. Two have currently been implemented to support evaluations of HARATIO: Repeat and Stop All (section 4.4.5.3). The Repeat action functions as a primitive looping structure and repeats any preceding actions in the script a set number of times. The Stop All action immediately stops execution of all running behaviour scripts on the parent object. Given the nature of these actions, the display of the script in working area is embellished with additional iconography to assist with communicating their effects. If a Repeat action is added to a script, all preceding actions will be given a small status icon to indicate that they will be performed more than once (Figure 4.23h). The icon matches that displayed within the repeat symbol. If a Stop All action is added to a script, any actions located after it will appear with a red 'X' overlay as they will not be executed and are therefore invalid in their current position. The overlay is like that described previously in the discussion of TA deletion. This approach was considered preferable compared with disallowing symbol placement after a Stop All action. The latter would conflict with the expected behaviour of symbol manipulation and could result in confusion. The example (Figure 4.23i) shows the invalid placement of a Turn Left action after a Stop All action. Repositioning this action before the Stop All action would cause it to become valid and the overlay to disappear.

While the editor handles the process of creating and modifying behaviour scripts, the Behaviour activity menu provides options for managing scripts at an object level and evaluating their operation. Buttons on the right side of the menu, Previous and

Next, facilitate cycling between scripts attached to the selected object (Figure 4.24a). The buttons are only enabled while their respective operations are applicable: for the Previous button, this is when the first script is loaded; for the Next button, it is when the maximum number of scripts has been attached to the object and the last is loaded. The radial menu core displays the sequence number (sn) of the loaded script along with the total number. In the menu graphic shown in Figure 4.24b, the loaded script is sequence number two and the total number attached to the object is three. Earlier iterations of HARATIO instead displayed the loaded script sequence number along with the maximum number that could be attached—five as mentioned in section 4.4.5.1—however, this was amended to the described format in response to experimental feedback (see section 6.1.3).

The behaviour identification at the beginning of the working area display corresponds to the loaded script number displayed in the menu core. If the sequence number equals the total number—that is, the loaded script is the last in the set—and the object has not reached the maximum number of scripts that may be attached to it, the Next button appears with (Add) appended to its label. Selecting the Next button in this state will cause a new behaviour script to be added to the object and then loaded (in template form) into the working area. The menu core will subsequently update to identify the new loaded script. The act of accessing different scripts remains consistent regardless of whether an existing script is loaded or a new script is created.

As with the Edit activity menu discussed in section 4.4.4, the bottom of the behaviour menu provides an option to delete the loaded behaviour script. This is intentionally placed away from non-destructive options to avoid accidental selection. Deleting a behaviour script will cause HARATIO to re-sequence all following behaviours accordingly such that $sn \rightarrow (sn - 1)$. The two buttons on the left of the menu assist the user with further evaluating scripts by providing options to translate and test them, respectively. These two options are discussed in the following subsections.

4.4.5.5 Behaviour Script Translation

HARATIO's behaviour script language prioritises simplicity. Accordingly, the overall script structure and use of graphical TA symbols have been selected with the intent of catering to those without programming skills. While these measures aid in providing a less intimidating experience compared with typing abstract statements into a text editor, there are still aspects of the language that may be initially ambiguous, such as the operation of certain TAs. Clearing up potential ambiguities by providing additional levels of assistance is something that would be beneficial to HARATIO's target audience. While a reference page built into the application, akin to a simplified API document, would be one possible solution, requiring users switch back-and-forth between editing and reference interfaces while remembering the information contained in each is not likely to be all that helpful.

Flip, a visual programming language intended to assist with the development of computational skills, similarly uses a visual structure of connected blocks to denote scripts. Notably, Flip operates in a bimodal manner, displaying a real time plain English translation of scripts in addition to the visual block representation. As described by Howland and Good (2015), this supplementary presentation is intended to help users “understand the meaning of what they create beyond the mere arrangement of blocks”. Such an approach

suits HARATIO's behaviour language well. Behaviour scripts already conform to a sentence-based structure, so a translation mostly involves expanded descriptions of each TA symbol. Configurable TAs can be further expanded to include their associated configuration details.

The Behaviour activity menu provides a toggle control that governs the visibility of translation for the loaded script (Figure 4.24c). In earlier iterations of HARATIO, toggling this control on would cause a translation panel to appear over the entire editor panel much in the same way the configuration and delete panels appear. Toggling the control off would cause the panel to disappear. This approach only permitted modifications to the behaviour when the panel was hidden and was later revised after conversations with users (see section 6.6.2.3) suggested it would be better placed below the working area. While the initial design provided adequate space for longer behaviour translations to wrap and be displayed in their entirety, the disconnect between working and translation panels was detrimental to useability as users would be required to constantly toggle translation on and off during editing. The placement of the panel below the working area permits both to appear concurrently and, as with Flip, allows constant evaluation of scripts in real time. To overcome translation overflow, the panel was made scrollable and linked to the working area such that scroll movement of one affects the relative scroll position of the other, and vice versa.

The presentation of script translation incorporates the when-do-end labels and maintains the colour scheme used in the working area: labels are displayed in white while TA translations appear in purple. The use of colour provides a clear separation between the component parts of the script and helps with identifying the words that correspond to specific TA symbols. As mentioned before, configurable TAs are translated with their associated configuration details. The translation of the first trigger in Figure 4.24d after the When label shows the configured Wait trigger appearing as 3 seconds have elapsed.

4.4.5.6 Testing

When developers create applications using modern Integrated Development Environments (IDE), they typically write and edit code without any of that code executing. It is not until the application is run (or debugged) that the code becomes active. This separation between contexts—*author* and *run*—similarly applies to HARATIO's Behaviour activity. Behaviour scripts remain dormant while being authored to prevent other scripts or objects interfering with their creation. Scripts only become active when the scene is *run*, which occurs when the associated marker is tracked in AR mode.

Although this approach provides a clear separation between authoring scripts and running them, there is a time cost associated with testing. The user must exit the Behaviour activity, switch to AR mode, and then reacquire the marker to view the script executing. If changes are required, the reverse steps must be performed to re-enter the Behaviour activity before any changes can be made. This adds up to a potentially frustrating and time consuming workflow.

To ameliorate this experience, the behaviour editor supports testing of individual behaviour scripts directly within the authoring environment. Initiating a test evaluates the loaded script as indicated by the sequence number displayed in the radial menu core. As triggers may involve the use of device sensors or AR tracker state, including them

in evaluation would be problematic and so only the script's actions are executed during a test run. Rather than filter trigger evaluation to a subset of those not dependent on such resources, all triggers are excluded from evaluation to avoid confusion. While this ultimately results in partial behaviour testing, the parts of the behaviour responsible for modifying object state (the actions) are available to evaluate in a quick and seamless fashion.

Users can test a behaviour script's actions by selecting the `Test Actions` button available on the Behaviour activity menu (Figure 4.24e). This causes the behaviour editor to temporarily disappear and the behaviour script's actions to be executed in sequence. The user may choose to stop a running test at any time by selecting a `Stop` button that appears in place of the `Test Actions` button whenever a test is in progress. Once all actions have finished executing, the behaviour editor will reappear to allow the user to make additional changes to the behaviour as necessary. While this feature does not provide exhaustive debugging or execute the script's triggers, it does enable users to evaluate the parts of their scripts that affect the state of an object in much less time than the alternative method. For novice users, this provides another feature that can be used in conjunction with script translation to aid in the clarification and comprehension of behaviour scripts.

4.5 Summary

This chapter has described HARATIO, a handheld AR authoring tool developed for the purposes of evaluating user interfaces and user experiences suitable for novice users. HARATIO incorporates AR usability findings relevant to modern handheld form factors (refer to Chapter 3) and provides a clear separation between live AR and author modes using a play-pause metaphor consistent with the operation of commonplace media playback controls. The author mode employs a freeze technique that stabilises the interaction space and enables authoring tasks to be completed without requiring the device constantly track a fiducial marker. The interface scales to suit various screen sizes.

An adaptation of a radial menu facilitates access to HARATIO's three core authoring activities: *create*, *edit*, and *behaviour*. The menu encompasses a dynamic hierarchical structure in which each distinct menu appears concentrically around a central core. The core serves to provide navigation controls and can additionally display supplementary information for various activity operations. The design respects screen space limitations common of mobile devices by displaying only one active menu at a time—the deepest child menu accessed. Parent menus automatically collapse to rings surrounding the core, communicating the current level within the hierarchy, and are embellished by a visual breadcrumb overlay that provides a relative indication of each menu item selected.

The Create activity enables virtual objects to be added to an AR scene. Object creation is assisted by a visual callout that clarifies object placement by revealing areas of the screen that would otherwise be occluded by users' fingers or hands. The Edit activity provides various options for manipulating existing virtual objects. The radial menu is used to support common rotate, scale, and translate operations in both direct and indirect capacities; the latter incorporates the use of dedicated controls within the menu. In addition to standard object colours, a dedicated occlusion colour is available that offers support for physical objects occluding virtual objects via a custom rendering approach. Finally, the Behaviour

activity allows users without programming experience to define object interactivity in the form of behaviour scripts. Scripts are created by arranging graphical symbols into a template structure that mimics a simple sentence phrase. Additional explanation of script intent is also accessible via a real time natural language translation that may be toggled to appear concurrently below the editor.

HARATIO has evolved to the state described in this chapter based on empirical data and user feedback obtained from two user evaluations, both of which are presented over the course of the following two chapters. Chapter 5 assesses overall user experience (usability) for a range of device form factors. Chapter 6 evaluates two key user interface elements—the radial menu and behaviour script editor—with a participant sample that includes novice users with no prior programming or HARATIO experience.

HARATIO User Experience Evaluation

This chapter presents a user experience evaluation of HARATIO across a range of handheld devices representing the form factor categories described in section 3.1. The evaluation was focused on the overall usability of the tool to determine whether the implemented design decisions resulted in an authoring experience usable for AR content creation. Aspects of the final version of HARATIO described in Chapter 4 were not available for this evaluation and were subsequently implemented based on results and feedback discussed in this chapter.

Tasks were intended to exercise all aspects of HARATIO's design, from locating and tracking a marker to transitioning to author mode and building a complete AR scene using the three available authoring activities. Compared with the study reported in Chapter 3, where existing tasks were evaluated by participants as end users, this study cast participants as authors and had them assume the content-creator role directly.

As HARATIO includes a freeze capability to address issues observed with the use of larger handheld devices, results between form factors were expected to be closer and, in some cases, better with larger screens. To thoroughly evaluate the in situ nature of HARATIO as a mobile AR authoring solution, the study design also incorporated a simulation of an office setup to compare to an in situ one. A key component of HARATIO's overall utility and usefulness as an authoring tool for novices would be in its ability to offer a viable alternative to technical tools. Such tools are typical of stationary office environments involving PCs, and these environments are often not collocated with the authoring location.

The remainder of this chapter begins by describing each of the experimental tasks and elaborating on the procedures used to evaluate them. The chapter then presents the results and discusses several user experience issues identified. These issues were subsequently addressed in the following chapter, to align with the final version of HARATIO in Chapter 4, before a further evaluation was performed to assess the suitability of key user interface components for novice users.

5.1 Tasks

The tasks created for the evaluation were designed to explore the overall user experience of HARATIO as an AR authoring solution. Structured and unstructured situations were devised in which participants either replicated an existing AR scene based on a strict set of requirements or were afforded the freedom to explore the authoring activities to design and compose their own solution to a scenario description.

5.1.1 Task 1

The first task shared similarities to task two from the end user evaluation of handheld form factors discussed in Chapter 3. The task involved a square plane object that was required

to move along a rectangular path around the centre of the associated marker. Participants were told that the object should move at a speed twice that of the default, begin moving as soon as the marker was tracked, and be coloured orange. Once the object was selected by tapping on it, it should respond by changing colour to green and stop moving. While the complementary task from Chapter 3 had the plane object loop around its path indefinitely, the implementation for this experiment was adapted such that it required the object only complete five laps. This was done to reduce complexity as it was anticipated participants would find it easier to comprehend and implement a set of behaviour actions that repeated a finite number of iterations rather than an infinite.

Successful completion of the task required participants first add a new plane object to the scene, change its colour to orange, and then configure two behaviour scripts. The first behaviour would need to move the plane along the appropriate path around the centre of the marker at the required speed. This would involve a combination of `Set Speed`, `Move Forward`, and `Turn Right` actions. The second behaviour would need to capture the required interaction by responding to a `Tap` trigger and then execute two actions: a `Colour` action to change the object's colour to green and a `Stop All` action to stop all running behaviours and halt the object's movement. For a description of behaviour triggers and actions, refer to sections 4.4.5.2 and 4.4.5.3.

Specific details (as described) including the size of the plane object, its starting location on the scene grid, the path it should follow around the marker, the speed of movement, as well as how it should respond to user interaction were all specified in an accompanying reference sheet that was provided to participants.

5.1.2 Task 2

The second task was much less structured than the first and was based on a written scenario exercise. Participants were given a marker and a reference sheet containing a description of the scenario and asked to author a suitable scene as best they could. The task was intended to let participants build on experience gained with HARATIO to pursue a more creative approach that would ideally involve further exploration of the interface and feature-set.

To aid in collecting reliable test data, Rubin and Chisnell (2008, p.183) suggest presenting a task scenario as a realistic situation with a believable motivation for why it is being completed; that is, something that makes it easier for participants to “stay in [the] role and overcome latent hesitation and self-consciousness.” The task two scenario was therefore presented as a fictional situation in which each participant was told they were part of a group of university students completing a project to build a mobile AR game. To help visualise the design of a specific level in the game, the other members of the group had tasked the participant with creating a quick AR mock-up complete with asset animation. The authoring workspace included physical objects that would need to be considered when designing the scene. The description of the level described a virtual tank object crossing a bridge, supported by a central pylon, between two cliffs and then passing under a nearby archway. The tank would begin its patrol path five seconds after the level was loaded (its associated marker tracked). The archway and one of the cliffs were represented as cardboard models, and each was placed in the workspace next to a

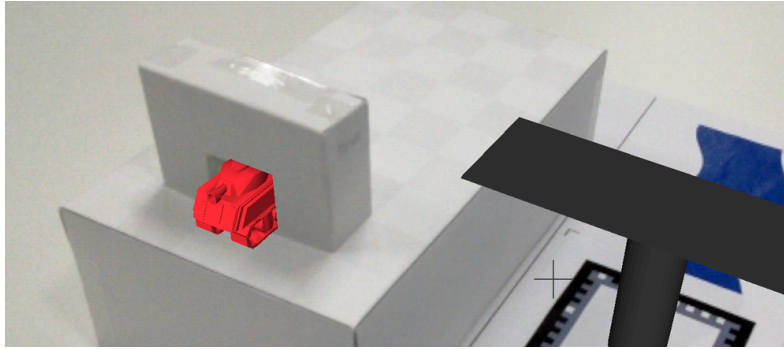


Figure 5.1: Occlusion between cardboard archway and tank object (shown in HARATIO).

printout of a river before the task was attempted. In addition to serving as a visual aid, the river printout was designed to assist with locating the cardboard models and correctly placing the marker—alignment guides indicating where the marker should be positioned were included. All other objects necessary to complete the task would need to be created by the participant as virtual objects. The reference sheet specified the kinds of objects participants should use for each component: a cube for the second cliff, a plane for the bridge, and a cylinder or cube for the bridge support pylon. The motion of the virtual tank over the bridge and under the archway would also need to be defined as one or more behaviour scripts. A few illustrations that depicted what the final scene could look like were included to help clarify the description.

Although not a requirement for completion, the accompanying reference sheet also included a hint reminding participants that virtual objects could be assigned an ‘occlusion colour’ to act as a virtual proxy for a physical object—in this case, each of the cardboard models. As discussed in section 4.4.4.1, the occlusion colour is rendered as a depth mask that enables physical objects to appear to correctly occlude virtual objects. In terms of the archway, participants would ideally build a virtual copy and assign it the occlusion colour. This would result in the tank appearing to be occluded as it passed underneath, as shown in Figure 5.1. While the concept of occlusion introduces technical complexity to a tool designed for novice users, it would be interesting to see how well HARATIO was able to convey this concept and whether participants understood the idea of an occlusion colour well enough to effectively implement it.

Rather than analyse how well participants could follow a set of instructions to replicate a scene, the assessment of task two was more concerned with the way participants approached the design process and the features of HARATIO utilised to create scene content. Nonetheless, successful completion of the task still required the creation of virtual objects appropriate for the scenario description as well as the definition of one or more behaviour scripts to enable the tank object to move across the bridge and pass under the archway. Specifics such as object location, colour, scale, and the behaviour algorithm used were left to the discretion of each participant.

5.2 Hardware

Four different devices were used for the evaluation: three handhelds and a desktop PC. The handheld devices comprised a smartphone, a mini tablet, and a tablet. The specific

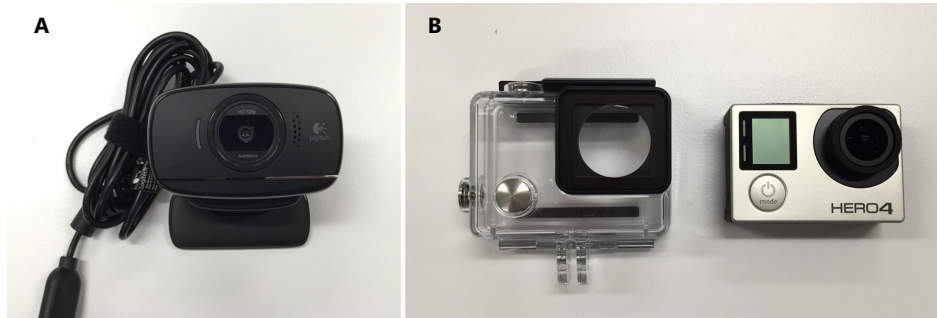


Figure 5.2: Logitech HD C525 webcam (a). GoPro camera and housing case (b)

handhelds used were identical to those discussed in Chapter 3. Detailed specifications and descriptions are provided in section 3.3 of this chapter.

A desktop PC was included to evaluate how users reacted to using HARATIO in a more traditional setting reminiscent of a typical office environment. The PC was an Alienware Area-51 (R2) configured with an Intel Core i7 5820K, 8 GB of DDR4 memory, 128 GB of SSD storage, a 2 TB hard drive, and an NVIDIA GeForce GTX 960 graphics card. The system was connected to a Dell U2713 27-inch monitor, and the operating system installed was Microsoft Windows 10 Home (64-bit). A Logitech HD C525 webcam (Figure 5.2a), affixed to a generic microphone stand, was positioned on top of a table and oriented face-down so it pointed towards the floor. The webcam provided camera data to the PC for AR tracking. As the development environment did not support the use of a webcam when HARATIO was deployed as a desktop application, PC trials were run directly within Unity. Although not ideal, this did not prove problematic as the environment was configured to run full screen and participants were not aware of or distracted by irrelevant parts of the Unity interface. Interaction with HARATIO running on a PC functioned as per the handheld devices with the exception of the mouse being used as a substitute for touch input. In reference to Figure 4.12 in Chapter 4, object selection was performed with a left-click, viewpoint orbiting with a left-click-and-drag, viewpoint panning with a right-click-and-drag, and viewpoint zooming with mouse wheel scrolling.

Finally, participants wore a head-mounted video camera while completing the tasks so interactions with the devices could be recorded from a first person perspective. The camera used was a GoPro Hero 4 Silver attached to a head strap via a GoPro housing case (Figure 5.2b). Recordings were made at a resolution of ‘1080p60’ using the ‘ultra-wide FOV’ setting. As the GoPro does not allow recording sans audio, videos were re-encoded post-experiment to remove the audio track and reduce file size.

5.3 Procedure

A total of fifteen participants were recruited from the School of Computer Science, Engineering, and Mathematics at Flinders University¹. All were male and currently studying at an undergraduate or postgraduate level. The age range was between 21 and 50.

¹ Any experiment involving human participants from Flinders University requires appropriate ethics approval from the university’s Social and Behavioural Research Ethics Committee (SBREC). Appendix D contains a copy of the final approval notice.

The study used a within-subjects design with each participant completing all tasks. The device form factor was the independent variable. To mitigate learning effects, each participant was assigned a counterbalanced device ordering beforehand that remained consistent throughout the experiment—they were not made aware of this order. Upon arriving at their scheduled time, participants were informed of the session format by the test moderator. A script was used to ensure each participant received the same information. They were then asked to complete a background questionnaire that included questions on handheld device ownership as well as experience with 3D modelling and visual programming. Out of the fifteen participants, twelve indicated owning a smartphone, six a tablet, and one a handheld game console. Except for the game console, all owners of smartphones and tablets used them for some form of content creation, such as writing emails or editing photos/videos. Eleven participants indicated experience with 3D modelling while only three had previously used a visual programming tool. Given most participants had a computer science background, a lack of visual programming experience was not surprising as they would instead make use of professional software (including text-based programming languages) throughout their tertiary studies.

During the experiment discussed in Chapter 3, participants were provided with an overview document that detailed the application interface and a method for completing each task. While this was intended to mitigate learning effects and be studied in detail prior to the experiment beginning, asking participants to completely absorb all of the document's content just before attempting the tasks proved overwhelming. In an effort to address the observed shortcomings of a document approach while still reduce learning penalties as much as possible, a brief training video was produced as an alternative. The video outlined the core features of HARATIO's user interface and was narrated by the test moderator. Seven key concepts were covered including how to pause and resume a scene, navigate the menu, move around the authoring environment, create new objects, edit existing objects, add occlusion support, and define object behaviours. The video lasted approximately eight minutes and participants were encouraged to seek clarification on any aspect they did not understand. Part of the video narration also included an explanation of AR if this was unknown to the participant; this generally occurred in conjunction with a demonstration of how to capture a marker in preparation for authoring.

Save for a one-on-one demonstration, the use of an orientation video was considered the most efficient and consistent way of illustrating the concepts of HARATIO that could be easily understood and absorbed. Compared to a document, the video presented application interactions occurring in real time and in a manner that was unambiguous. A hands on training session was dismissed to avoid influencing participants' impressions of the tool before the test: their reactions and experiences when using it for the first time were important.

Following the video, participants were instructed to complete the first task. Prior to beginning, they were shown a video of the task in its final state and could view the expected interactions. They were then given a handheld device (as per their pre-assigned device ordering), a marker, a task reference sheet containing the details necessary to complete the task (provided in Appendix F), and told they were free to adopt any comfortable device pose during authoring. An illustration of the task materials arranged on a table is provided in Figure 5.3. At this point, participants were also asked to wear the head-mounted camera.

The camera was adjusted to provide a first person perspective of the device and capture their interactions with it. Although wearing the camera was not a requirement, all who took part agreed without objection. In addition to recording interactions via the head-mounted camera, the screen of the device was also captured while the task was attempted². This provided a redundancy in case a participant objected to wearing the camera, the recording failed, or it was moved unintentionally. Participants were told they could remove the head-mounted camera once they had concluded their attempt.

Following completion of the task, participants were asked for a rating of performance satisfaction in relation to using HARATIO on the current device. The rating was recorded as an answer to the Single Ease Question (SEQ) (Sauro and Lewis, 2012, p.214). The SEQ comprises a single question and asks users to rate how easy or difficult a task was complete. Responses were given on a seven-point Likert scale with one representing 'very difficult' and five 'very easy'. As with the previous experiment discussed in Chapter 3, responses were provided verbally following a printed scale being held up.

Participants were then asked to complete the second task using the same device. As the second task was scenario-based, they were not shown a video preview of the completed scene. Instead, they were given a marker and a task reference sheet (provided in Appendix G) and instructed to author an appropriate scene at a specific location within the testing environment. This location included the cardboard models that needed to be considered when designing the scene. An illustration of the task materials is presented in Figure 5.4. Participants were asked not to move the cardboard models and instead create and position virtual objects around them. As with task one, the head-mounted camera was worn for the duration of the task and a verbal SEQ rating sought following its completion.

Once both tasks were finished, a System Usability Scale (SUS) (Brooke et al., 1996) and Handheld Augmented Reality Usability Scale (HARUS) (Santos et al., 2014, 2015) questionnaire were completed to gather feedback on the overall usability of HARATIO with respect to the current device. The HARUS shares similarities with the SUS but emphasises perceptual and ergonomic factors specific to handheld AR. SUS questions were as per Brooke's original phraseology save for the replacement of the word *system* with the term *authoring tool*, to better reflect what was being assessed, and the word *cumbersome* in question eight with the word *awkward*. The decision to replace the word cumbersome was informed by suggestions from Finstad (2006) and Bangor et al. (2008), who independently discovered a number of participants would seek clarification over its meaning, especially those whose native language was not English. As volunteers were recruited from within Flinders University, a multi-cultural institution with many foreign students, such a change seemed appropriate to incorporate. Both questionnaires were answered by indicating the level of agreement or disagreement to various statements on a Likert scale. As Brooke noted when describing the SUS, questions alternate between being positively worded and negatively worded to mitigate response bias and encourage participants to think about their answers. The SUS and HARUS were compiled into a single long-form questionnaire, the ordering of which was also counterbalanced between participants. SUS questions were recorded on a seven-point Likert scale (instead of the original five) to match the scale used by the HARUS. Finstad (2010) evaluated five- and seven-point versions of the SUS

² Screen recording was accomplished using the Android application 'Unlimited Screen Recorder' (Zdziebło, 2015). The application was configured to record screen touches with no audio.

and concluded the seven-point scale to be less subject to response interpolation and more 'likely to reflect the respondent's true subjective evaluation'. Following completion of the questionnaires, the described process was repeated with the remaining two handheld devices.

After all devices had been used, the second task was completed one last time using a desktop PC. The PC was included as a means of contrasting the use of HARATIO in situ on a mobile device versus a (simulated) typical office environment—it didn't make sense to have participants use it to complete the first task as this was location-independent. The PC was separated from the authoring environment and required that participants move between the two locations to author and preview their work. Each time a participant elected to move between locations, the test moderator would reposition the webcam for an overview of the alternate location. The decision to separate environments was done to simulate the case where the authoring location and authoring device are not collocated, which would be common in typical office environments involving a PC. This would allow participants to get a sense for whether having an AR authoring tool available in a mobile, self-contained package would be beneficial to the authoring workflow. They would have the opportunity to formally comment on this in the post-experiment interview. As the office environment for the PC was simulated, however, the distance between it and the authoring location was not as far apart as in a real world setting. Consequently, the tangible effects of the separation were limited. Consistent with all previous attempts, a post-task SEQ rating was collected and participants completed a final SUS questionnaire.

To conclude the experiment, a brief interview was conducted in which participants were given the opportunity to expand on their questionnaire ratings, highlight any issues encountered with HARATIO, and provide suggestions for improvements. The interview comprised eight questions and was digitally audio recorded (with consent) for later transcription.

5.4 Hypotheses

The experiment tested three hypotheses that explored factors relevant to the user experience of HARATIO as a mobile AR authoring tool across various form factors:

- H1. HARATIO achieves a mean usability score with SUS and HARUS instruments of at least 70.14 (average score for a passable system) on smartphone, mini tablet, and tablet form factors.
- H2. With the implementation of a freeze technique to stabilise the interaction space, the tablet form factor matches or exceeds the task completion rate and task-performance satisfaction (SEQ) ranking achieved by the smartphone.
- H3. Authoring content with HARATIO is more efficient on larger devices. As screen size increases, the time required to author an AR scene decreases.

5.5 Results

This section presents empirical results and corresponding analysis of participant data obtained from task one and two. The subsections that follow address the hypotheses

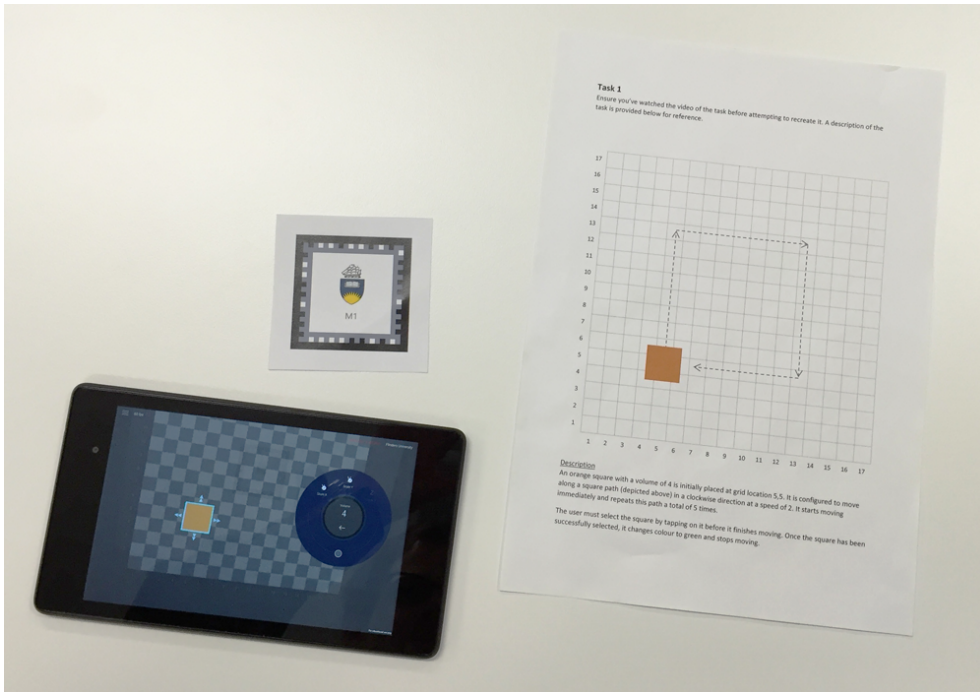


Figure 5.3: Testing materials used for task one: fiducial marker, task reference sheet, and handheld device (mini tablet).

Marker alignment guide

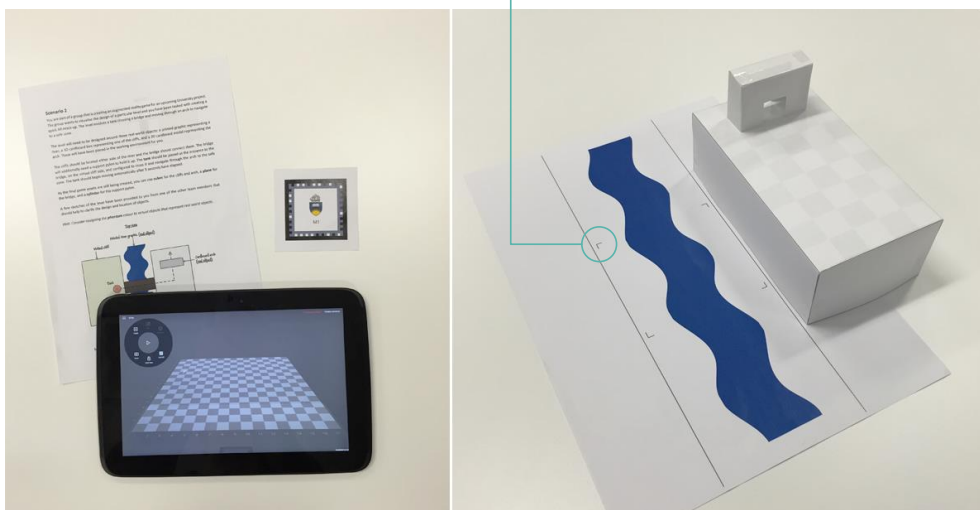


Figure 5.4: Testing materials used for task two. Left: task reference sheet, fiducial marker, and handheld device (tablet). Right: river printout (marker alignment sheet) and physical cardboard objects.

listed in section 5.4 and discuss responses from the post-experiment interview. Interview questions provided participants with the opportunity to comment on the use of HARATIO and any particular difficulties encountered.

5.5.1 Task Completion Time

Task completion times were determined using log data and calculated from the participant's first interaction with HARATIO following the scene being paused to completion of the final interaction before handing the device back to the test moderator. As the sample population was less than 25, task times were analysed using the geometric mean (GM) rather than the sample median. As reported by Sauro and Lewis (2010), the GM provides a better overall estimate of the central value for small samples with less error and bias.

Completion times for both tasks are presented in Figure 5.5. Overall, participants completed the first task in the least amount of time when using HARATIO on the mini tablet, recording a GM of 181 s ($SD = 238$ s). This increased to 209 s ($SD = 325$ s) when using the tablet and further increased to 235 s ($SD = 222$ s) when using the smartphone. Using HARATIO to complete the second task (scenario exercise) was quickest on the smartphone with a GM of 490 s ($SD = 256$ s), followed by the tablet at 503 s ($SD = 308$ s), and finally the mini tablet at 555 s ($SD = 439$ s). In contrast to task one, participants took longer completing task two with all form factors save for three attempts.

The results do not indicate a clear trend in terms of authoring time increasing or decreasing with HARATIO when moving between adjacent form factors. To further investigate the data and test H3, a one-way repeated measures ANOVA was performed on completion times for each task at the $p < 0.05$ level. Both task one ($F(2, 28) = .39, p = .682$) and task two ($F(2, 28) = .69, p = .511$) results were insignificant and indicate the time required to complete the tasks with HARATIO was not influenced by form factor. Therefore, H3 is rejected.

5.5.2 Task Completion Rate

Evaluating task completion involved analysing a combination of log data and video recordings captured of participants' interactions with HARATIO. Task success was graded using a series of metrics that determined how much of the task had been completed according to the requirements. As discussed, the first task had a stricter set of requirements compared with the second as the latter was intended to be less structured and open to more creative flexibility. A partial success method was adopted in favour of a binary pass/fail approach so as not to unfairly dismiss attempts that contained minor errors. Aspects of each task that were discovered to be a frequent source of problems were further analysed to evaluate their interpretation and effect on the use of HARATIO to complete the associated requirement.

5.5.2.1 Task 1

The metrics used to assess task one are presented below and correspond to the task requirements. Each metric itemised with a letter prefix was worth one point. Given the adoption of a partial scoring method, participants could therefore be awarded anywhere between 0 and 11 points for their attempt.

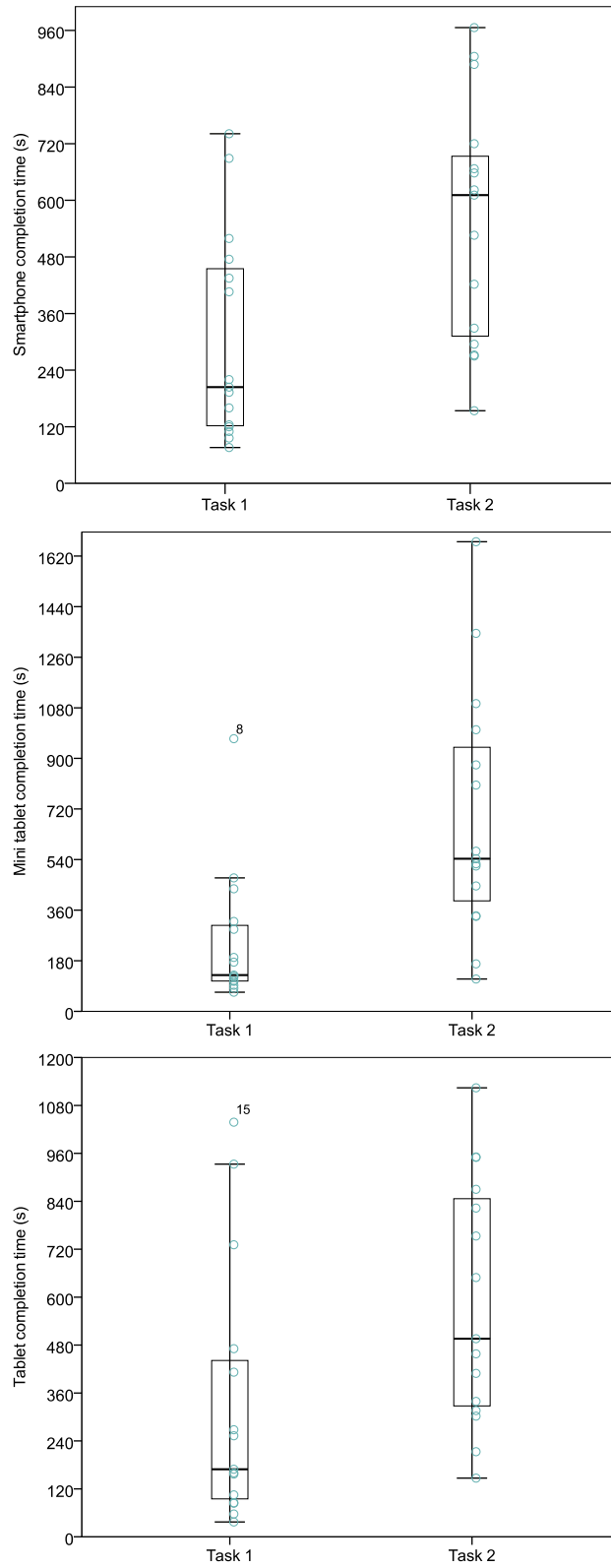


Figure 5.5: Task one and two completion times for the use of HARATIO on each form factor

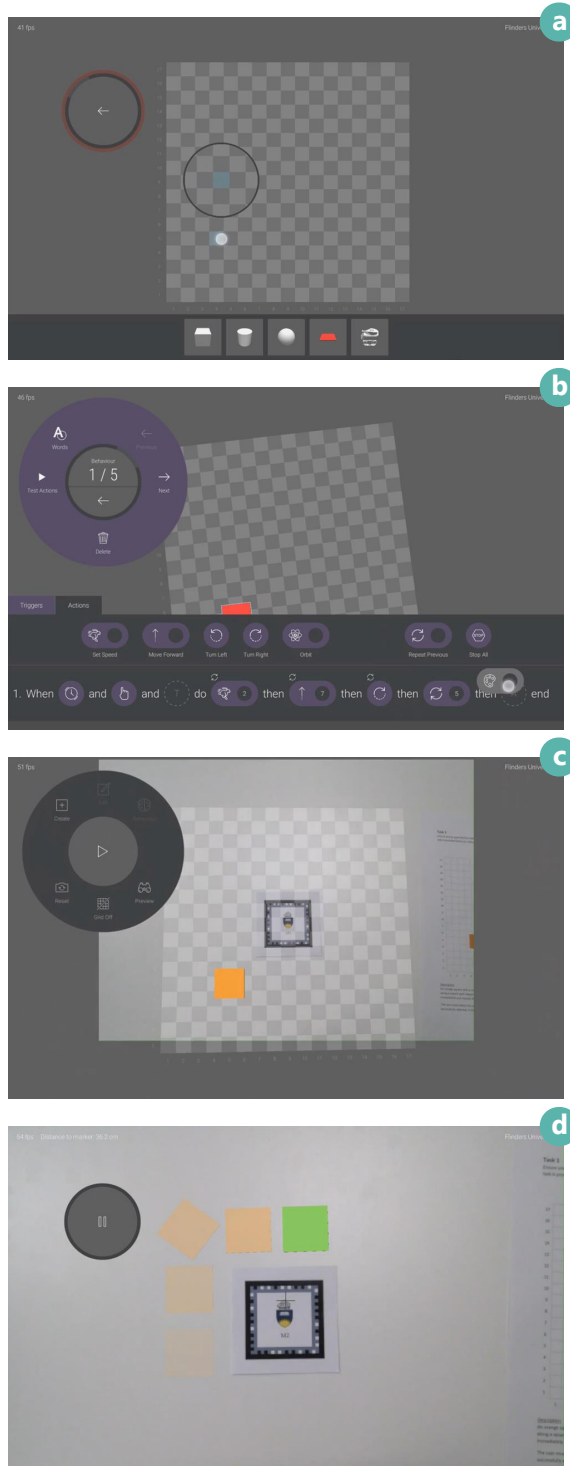


Figure 5.6: Screen captures of task one attempts. All images taken from mini tablet recordings.

1. Object creation
 - (a) Added a suitable object to the scene: a plane or cube was accepted
 - (b) Positioned the object at starting grid location (5, 5)
 - (c) Scaled the object to a volume of 4 (or 8 if a cube was used)
 - (d) Coloured the object orange
2. Behaviour one definition
 - (a) Assigned a Now trigger to begin executing actions immediately
 - (b) Assigned a Speed action with a value of 2
 - (c) Assigned appropriate combination of Move Forward and Turn Right actions to move the object along a clockwise square path
 - (d) Repeated path 5 times
3. Behaviour two definition
 - (a) Assigned a Tap trigger to begin executing actions when the object is tapped
 - (b) Assigned a Colour action with a colour value of green
 - (c) Assigned a Stop All action to stop all running behaviours on the object

Overall completion rate as an aggregate of 45 attempts—15 participants each used 3 handheld devices—was 79%. By device form factor, participants achieved a mean task completion score of 8.73 ($SD = 2.19$) with the tablet (79%), 8.7 ($SD = 2.02$) with the smartphone (79%), and 8.53 ($SD = 2.17$) with the mini tablet (78%). Similar results were achieved for all form factors with the smartphone and tablet returning remarkably close scores save for a minor difference in variation. As the tablet was able to match the smartphone in completion performance, the task one results support H2. Example screen captures from recordings of participant attempts are provided in Figure 5.6. In each image, the task is shown in a different state of completion that ranges from object creation (a), behaviour script definition (b), re-entering author mode to correct a minor mistake (c), and checking the final sequence of the completed scene (c). In the case of (c), several images have been combined to better present the movement of the plane up until the moment of selection.

In addition to the overall outcome, results for each metric category were also analysed independently. For object creation, the data revealed 22 (49%) of the 45 attempts were

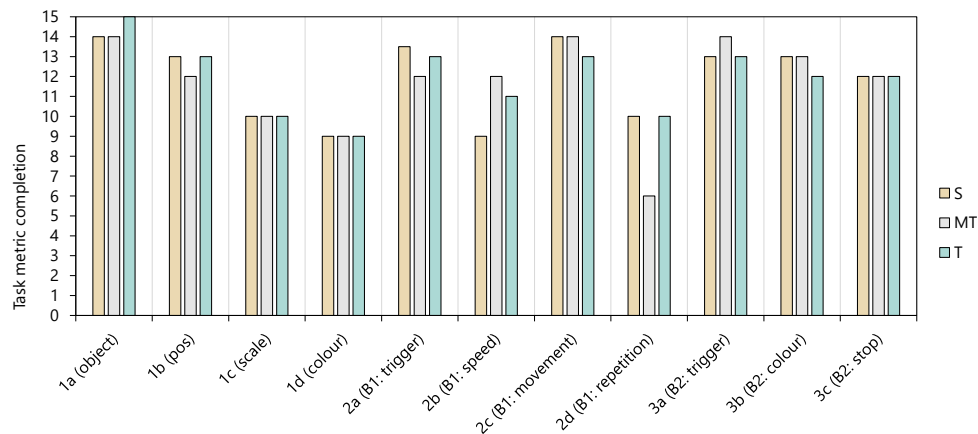


Figure 5.7: Completion of task one metrics by form factor

successful in completing all requirements (1a–1d), while for the two behaviours, all necessary components—correctly configured triggers and actions—were implemented in 14 (31%) attempts for the first behaviour (2a–2d) and 36 (80%) for the second (3a–3c). The breakdown of metric categories shows issues were predominately experienced when creating the object (1) and configuring the associated behaviour to move it along the square path (2).

As shown in Figure 5.7, incorrect colour and incorrect scale comprised most of the issues encountered with object creation. In the cases where these were not implemented correctly, further investigation revealed the participants simply left the colour and scale values at their default rather than setting them incorrectly. This sometimes occurred with all three devices while other times it only happened with one or two. It could be surmised that the related requirements in the task reference sheet weren't fully understood, were overlooked, or the participant simply forgot to act on them. Given participants were asked to repeat the same task multiple times, the latter is plausible. The distribution of successful configuration of object scale and colour were identical between form factors, however, indicating the difficulties were not device-dependent. This adds further support to H2.

The dominant issues encountered with the first behaviour were related to the configuration of path repetition (2d) and movement speed (2b). In the case of movement speed, examination of the data revealed the participants either forgot to include it in their script or misinterpreted the required value and set it to one. As a value of one represented the base speed, this didn't result in any change occurring. Investigation into path repetition issues also suggested a contributing interface issue with the behaviour editor. Other than the required functionality, specific behaviour script configuration was not explicitly stated in the task reference sheet as multiple approaches could be taken to achieve the same outcome. The simplest configuration would involve creating several instances of `Move Forward` and `Turn Right` actions until the desired path movement was achieved. Efficiency and readability could be improved by defining the base movement actions and then adding a `Repeat` action to replicate movement as required. Analysis of video recordings suggested the implementation of the `Repeat` action was a source of confusion as its configuration value (the number of repetitions) appears to have been taken literally; that is, the configured value was assumed to represent the total number of iterations rather than the number of repetitions of previous actions—a subtle but important distinction. A similar misinterpretation was observed during initial pilot tests of HARATIO's behaviour editor and led to a design revision of the `Repeat` action configuration panel to clarify the action's operation. The hint message displayed at the top of the panel, shown in Figure 5.8, was amended to include a sentence that read: *With a selection of x, previous actions will run a total of y times.*

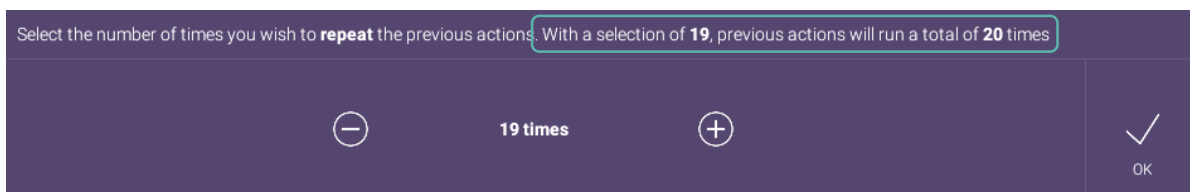


Figure 5.8: Clarification of hint message in `Repeat` action configuration panel

The value of x reflected the action's configurable value as set by the user while the value of y reflected the total number of repetitions (i. e. $x + 1$). Evidently, this change didn't provide sufficient clarification and was likely overlooked as users focused on the large configurable value in the centre of the panel with a 'how many times will this run?' mindset. Given this observation, a better implementation would have involved an amendment to the configurable value format to clearly indicate the total number of iterations rather than the number of repetitions. Further analysis of score distribution between devices reveals the mini tablet was the form factor participants struggled with this the most (Figure 5.7); the determination as to whether this was a result of coincidence or some aspect of the mini tablet hardware used in conjunction with HARATIO is unclear and presents an area for further investigation.

The behaviour script translation feature (see section 4.4.5.5) was designed to assist with clarifying situations of script ambiguity. As the symbols and labels used to represent individual triggers and actions might be vague to some users, displaying expanded descriptions of each in the form of a plain language sentence was intended to help clarify the intent of a behaviour in an unambiguous manner. For the Repeat action, the associated translation would have clearly stated that previous actions would be repeated n times rather than execute a total of n times. Out of the 45 attempts across the three form factors, the translation feature was only invoked on 12 occasions by 7 different participants; in each of these cases, it was only accessed once per attempt. The location of the menu item on the Behaviour activity menu to facilitate toggling the translation panel was pointed out during the training video, but given participants were using the interface for the first time and were expected to recall the various features available within each of HARATIO's authoring activities, it is conceivable they simply overlooked the menu item or didn't consider using it. It is also conceivable the labelling used was not clear enough in communicating the associated action.

5.5.2.2 Task 2

Metrics used for task two analysis are presented below. As the task was based on a scenario exercise, the metrics focussed primarily on the creation of necessary objects and the definition of a behaviour script to move the virtual tank. Specific object properties such as location, colour, and scale were left up to the discretion of each participant. The use of the occlusion colour was not necessary for task completion but was nevertheless counted to assess its use. As before, a partial scoring method was used with all but the last metric itemised with a letter prefix worth one point. Successfully addressing all scenario requirements would therefore yield a score of 6 points.

1. Object creation
 - (a) Created cube object to represent virtual cliff
 - (b) Created plane object to represent bridge
 - (c) Created cylinder or cube object to represent bridge support pylon
 - (d) Created tank object
2. Tank behaviour definition
 - (a) Assigned Delay trigger with a value of 5

- (b) Assigned appropriate combination of Move Forward and Turn Left actions to move the tank across the bridge and pass under the archway

3. Occlusion colour

- (a) Created occlusion-coloured objects to represent cardboard cliff and/or archway

Overall completion rate as an aggregate of 60 attempts—the PC was included as a fourth device type in addition to the three handhelds—was 78%. By device form factor, participants achieved a mean task completion score of 4.93 ($SD = 1.28$) with the PC (82%), 4.73 ($SD = 1.28$) with the smartphone (79%), 4.67 ($SD = 1.35$) with the tablet (78%), and 4.33 ($SD = 1.59$) with the mini tablet (72%). The scores are, again, similar between handheld form factors with the smartphone and tablet remaining close. The 1% discrepancy between completion score in favour of the smartphone does, however, oppose H2, although the difference between the total number of task points achieved with each form factor was only one: 71 for the smartphone versus 70 for the tablet.

Example screen captures of participant scenes for the task are presented in Figure 5.10. As before, the images depict the task in various stages of completion. Image (a) shows a participant checking the alignment of virtual objects with respect to real world scene assets, image (b) shows the tank’s behaviour script being defined, image (c) shows a participant previewing their scene in HARATIO’s AR mode to validate the tank behaviour, and image (d) shows a final scene incorporating the use of the occlusion colour. The virtual tank in the last image can be seen partially occluded by the cardboard archway as it begins to pass underneath.

Further analysis of individual metrics revealed aspects of the task that participants found most difficult. The graph in Figure 5.9 shows the bridge support pylon (1c) being the cause of most difficulties with the object only being created in 41 (68%) attempts. Analysis of video recordings as well as application logs reveals that in the other 19 (32%) attempts, participants omitted creating it entirely. As it was possible for the plane (bridge) to be directly translated on top of the cube (cliff), omission of the pylon would not have impeded task progress.

In terms of the tank behaviour definition, most difficulties were related to the trigger rather than the actions. Several participants created a valid behaviour script but assigned an

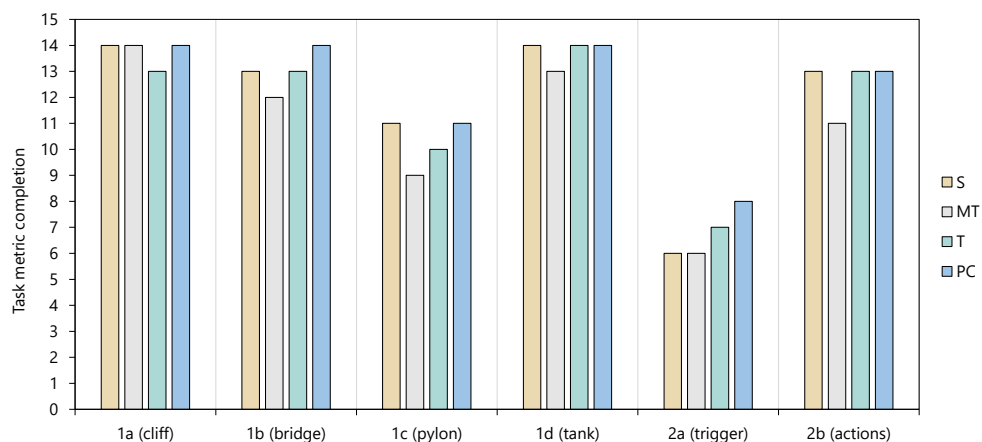


Figure 5.9: Completion of task two metrics by form factor

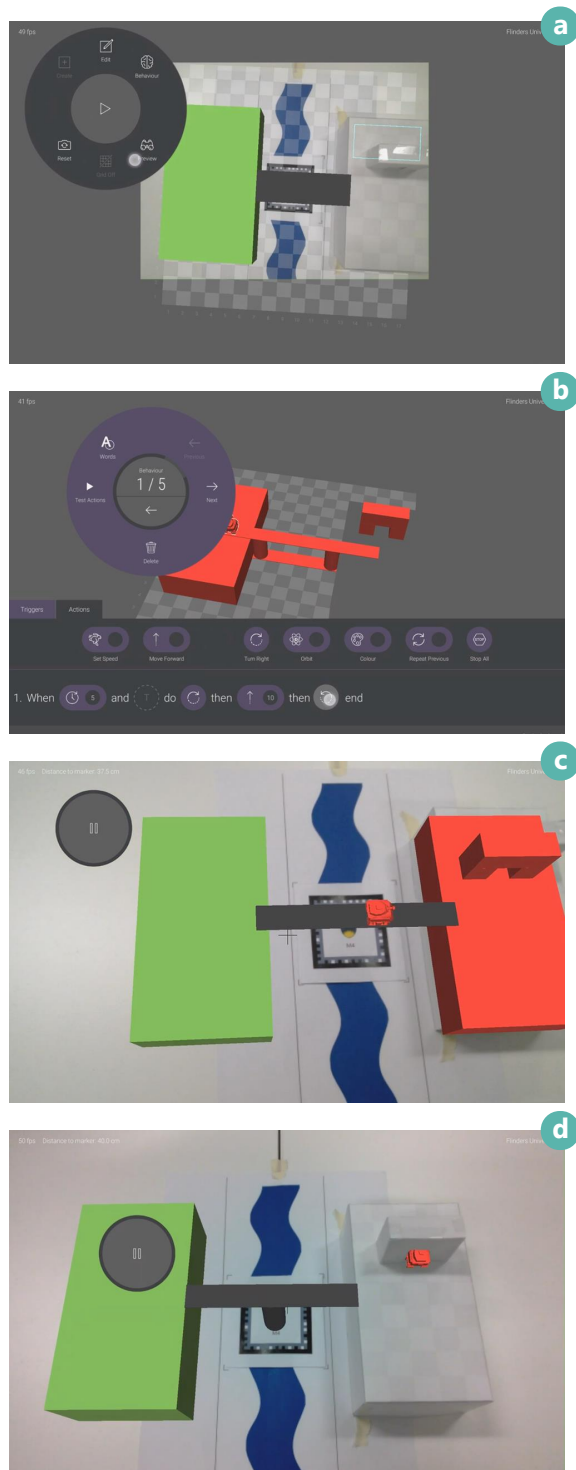


Figure 5.10: Screen captures of task two attempts. All images taken from mini tablet recordings.

incorrect trigger that either started execution immediately (via a Now trigger) or required the device first be focused over the tank (via a Focus trigger). In these instances, it is suspected that participants were either unsure as to which trigger would achieve the desired result or simply misread the trigger requirement when parsing the task reference sheet. Similar to the task one discussion, the former could have been aided by the use of script translation as expanded trigger descriptions would have provided clarification and highlighted instances where the functionality of a trigger was misunderstood. Out of the 15 participants and 60 attempts, however, only one participant was recorded as making use of it. This contrasts with the seven participants from the previous task. Interestingly, the participant who did use the feature did so for the first time; they did not access it during task one. It is possible script translation was not considered particularly beneficial or useful for completing the tasks; however, improving the discoverability of the associated menu item and clarifying the labelling could help promote its use. Re-evaluating the presentation of the translation panel is also worth considering. Permanently displaying the panel alongside the active behaviour script would solve any discoverability issues and surface the utility of the feature to more users. In addition to those who assigned incorrect triggers, 10 attempts from 6 different participants contained no behaviour configuration for the virtual tank. In these cases, the participants in question spent all their time creating and arranging objects and simply ran out of time.

For all metrics, the PC equalled or bested the handheld form factors even with the requirement that participants move between physical locations to author and preview their work. As the PC was always used last, its score could not be compared equally with the handheld devices due to learning effects; however, the results do indicate that HARATIO's user experience is not degraded when using mouse input.

Use of the occlusion colour wasn't counted towards task scoring, but its use was recorded as a measure of how well participants were able to understand and implement the concept of physical-virtual object occlusion. In total, 38 attempts (63%) by 11 different participants included at least one virtual object assigned the occlusion colour. Instances of only one object were almost always as per Figure 5.10b in which a proxy of the cardboard cliff was created. Seven of the 11 additionally modelled a proxy of the cardboard archway and produced a final scene in which the virtual tank appeared correctly occluded as it passed underneath (illustrated in Figure 5.10d). Such a high usage rate was somewhat unexpected but, for those that did experiment with occlusion, it demonstrates a conceptual understanding of the relationship between physical and virtual objects and the way in which the two should appear to coexist within the same scene. Additional discussion on the use of the occlusion colour is provided in section 5.5.6, which reports on participant responses to post-experiment interview questions.

5.5.3 SEQ

Completion of each task-device combination was followed by a verbal response to the SEQ on a seven-point Likert scale. The smartphone recorded the lowest overall rating with a mean of 5.07 ($SD = 1.33$). The smaller screen was likely to have presented increased interaction challenges and more instances of screen occlusion when displaying multiple elements from HARATIO's interface. The tablet received a marginally higher score with a

mean of 5.2 ($SD = .77$). This supports H2 and indicates the issues involving tablet weight and ergonomics, as described in Chapter 3, were not so overwhelming as to dominate the experience. The mini tablet proved the form factor with which participants found the task easiest and achieved a mean score of 5.87 ($SD = 1.13$). This suggests participants found the mini tablet to provide an acceptable compromise between the physical attributes of its adjacent form factors, enabling the task to be completed most effectively. As concluded in Chapter 3, the mini tablet form factor appears to be somewhat of a ‘goldilocks’ size for handheld AR as it offers a suitable balance between screen size and mobility. Overall, task performance satisfaction with all form factors was comparable to the average SEQ score (4.8–5.1) reported by Sauro (2012) from data comprising over 200 tasks and 5000 respondents. Based on these scores, participants found completing the task with HARATIO similarly achievable regardless of device form factor

The second task introduced the PC as a fourth device and evaluated HARATIO using a scenario exercise that was designed to be representative of a typical AR authoring situation. The PC is still a dominant tool in creative industries and offers more precise input techniques compared to a handheld, so its inclusion enables limited comparisons between collocated authoring and separated authoring in a (simulated) office environment.

Unsurprisingly, given task two was less structured, requiring more thought and independent interface exploration, participants rated it more difficult. Consequently, all three handhelds received below average SEQ scores. The standing between the handhelds remained unchanged, though the range of scores slightly increased from 5 to 6. The smartphone achieved a mean of 4.47 ($SD = 1.51$), the mini tablet a mean of 4.67 ($SD = 1.5$), and the tablet a mean of 4.6 ($SD = .91$). As with task one, participants found the mini tablet easiest for completing the task. The tablet once again bettered the smartphone with its lower score variance compared with other form factors suggesting a degree of consensus among participants. Except for the minor difference between task two completion scores discussed in section 5.5.2, the tablet has consistently supported H2. The intention of the hypothesis was to determine whether the addition of HARATIO’s freeze feature could alleviate the previously discovered ergonomic issues with the use of larger devices for AR. The results demonstrate a distinct improvement in performance and participant acceptance of the tablet as a viable form factor option. In light of this, H2 can be considered confirmed.

The PC scored distinctly higher than all handhelds with a mean of 6.2 ($SD = .68$). Even with the added burden of having to move between two locations to author and preview scene content, participants considered the PC preferable over all handhelds in terms of task easiness and performance satisfaction. The position of the PC among device ordering (last), the level of familiarity with mouse interaction, and the appreciably larger screen are suspected to have all contributed to this result. These assumptions were tested during the post-experiment interview, which is discussed in section 5.5.6.

5.5.4 SUS

Following the completion of both tasks with each device, participants were administered a SUS questionnaire in which they rated HARATIO’s usability in the context of the device just used. Separate SUS questionnaires were employed so comparisons could be made between form factors. SUS questionnaires yield a usability score between 0 and 100, which

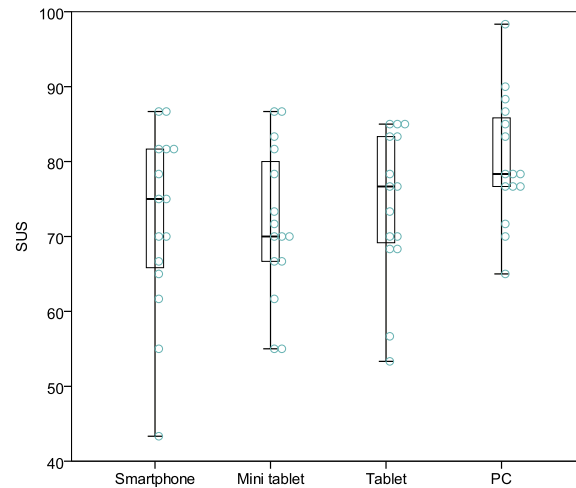


Figure 5.11: Participant SUS responses for smartphone, mini tablet, tablet, and PC devices

is calculated using the formula presented in equation 3.1. If device form factor played little part in the usability of HARATIO, the scores would be expected to be similar and support H1.

Figure 5.11 shows the SUS scores for each device as rated by participants. From the response data collected, handheld form factor did not appear to have any appreciable effect on usability score with similar means recorded for all handhelds: 74.22 ($SD = 9.98$) for the tablet, 71.89 ($SD = 12.16$) for the smartphone, and 71.78 ($SD = 10.17$) for the mini tablet. To further investigate, a one-way repeated measures ANOVA was performed on the SUS results at the $p < 0.05$ level to test the assertion that handheld device form factor does not affect the usability of HARATIO. The results revealed an insignificant result ($F(2, 28) = .72, p = .497$) and indicate participants found HARATIO to be similarly usable on all form factors.

The PC SUS score was recorded for completeness and as a supporting measure to participant comments in the post-experiment interview. Like the SEQ results, the PC bettered the handhelds with a mean score of 80.22 ($SD = 8.54$). Given the PC was always the last device used after all the handhelds, and only for task two, its higher score may have been influenced by the experience gained with HARATIO and the familiarity of the task at this point. However, despite the fact HARATIO was first and foremost developed for handheld devices, it is very usable on PC systems using mouse interaction in lieu of touch.

Though Brooke et al. (1996) never offered a formal rating scale for SUS scores, a few have been proposed based on analysis of large samples of SUS data. Bangor et al. (2008) discuss two rating systems referred to as the ‘adjective rating scale’ and ‘university grade analogue’ (Figure 5.12). The adjective rating scale is based on data collected from an additional seven-point Likert question appended to the SUS (Bangor et al., 2009). The question captures an adjective rating of a system’s user friendliness by asking users to rate it from worst to best imaginable. Bangor and colleagues found results from the question strongly correlated with SUS score and consequently provided a useful subjective label of usability. The mean scores achieved by all form factors equate to an adjective rating of *good*, with the smartphone and mini tablet falling just to the left of the indicated rating line and the tablet just to the right. The PC score of 80.22, for comparison, moves it closer

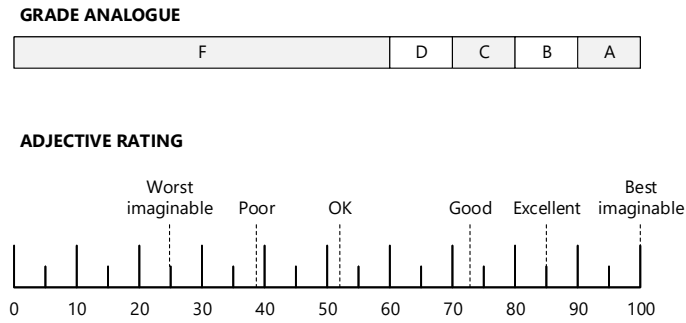


Figure 5.12: SUS rating scales: university grade analogue (top) and adjective rating scale (bottom) (Bangor et al., 2009)

to an *excellent* rating.

Alternatively, the grade analogue approach leverages the SUS score range of 0 to 100 and assigns an associated letter grade—A through D and F—at intervals mirroring typical academic grading. As a grade, the use of HARATIO across all handheld devices was a C. With the PC, it was a B.

Sauro and Lewis (2012, pp.202–203) proposed a slight modification to the scale by placing a score of 68 (a C) at the centre instead of 50; the former referred to the mean score obtained from analysis of over 5000 SUS responses. By contrast, Bangor and colleagues’ (2008) analysis was based on 2324 responses where the mean score was 70.14. A smaller factor analysis study of the SUS by Lewis and Sauro (2009) produced yet another result with a mean of 62.1 based on data from 324 responses. In pursuit of a benchmark to compare HARATIO’s performance against, the larger mean of 70.14 was selected on the basis it encompassed the two smaller ones.

Bangor et al. reason a passable system needs to achieve a SUS score of at least 70—in line with their observed mean—with better systems scoring in the high 70s to 80s and superior systems bettering 90. Based on this and the aforementioned benchmark, HARATIO can be considered passable or better on all devices tested, thereby supporting H1. While the key value in the SUS lies in the comparisons between form factors, the assignment of a label or grade to individual scores is also useful as a means of easily communicating overall usability to those unfamiliar with the SUS instrument. This was beneficial when discussing the work with supervisors, colleagues, peers, and interested parties.

With the noted exception of those who think they perform well in a task but do not, Bangor et al. suggest SUS score may be correlated with task performance. In the case of the data recorded for participant use of HARATIO, a Kendall’s tau-b correlation between SUS score and task completion rate for each SUS–task combination was performed to investigate whether such a relationship existed. The results revealed a small positive relationship between SUS and task one completion rate ($T_b = .239, p = .034$) that was statistically significant but almost no relationship between SUS and task two completion rate ($T_b = .065, p = .576$). Though small, the task one results support participants’ ratings of HARATIO’s usability. A contributing factor in the lack of task two correlation is likely the unstructured nature of the task: participants created their own solutions to a provided scenario rather than recreating an existing scene by adhering to a defined set of criteria

(as per task one). In this case, it would be reasonable to expect participants to consider they performed well and completed the task scenario correctly.

5.5.5 HARUS

While the SUS is a general usability instrument that can be applied to many different types of systems, the HARUS is specifically tailored to evaluate the suitability of handheld AR applications by considering perceptual and ergonomic factors. Participants were administered the HARUS in conjunction with the SUS following the use of each handheld device. The HARUS produces a final score on the same scale as the SUS (0 to 100) and is calculated using the formula in equation 5.1. The score contribution (H) from each of the sixteen questions (i) is summed and divided by 0.96. The contribution from odd numbered questions is the response minus one while for even numbered questions it is seven minus the response.

$$\frac{\sum_{i=2,4,\dots,n}^{n=8} (H_i - 1) + \sum_{i=1,3,\dots,n}^{n=8} (7 - H_i)}{0.96} \quad (5.1)$$

HARUS scores for each form factor are presented in Figure 5.13. The mini tablet achieved a mean score of 75.56 ($SD = 9.33$), the smartphone 73.68 ($SD = 11.62$), and the tablet 72.36 ($SD = 10.32$). Compared to the SUS discussed previously, the HARUS produced similar results. The mini tablet moved from last to first place in the device ordering, swapping places with the tablet, while the smartphone remained second. As the HARUS is a recent questionnaire and has limited data available from other studies to compare against or determine a benchmark, the SUS benchmark of 70.14, as described in section 5.5.4, was used as the performance metric of acceptability. A Pearson correlation performed between HARUS and SUS scores revealed a positive relationship for all three form factors that was statistically significant, supporting this decision. The tablet recorded the strongest relationship ($r(15) = .575, p = .025$) followed by the mini tablet ($r(15) = .573, p = .026$) and the smartphone ($r(15) = .529, p = .043$). Santos et al. (2014) similarly discovered a significant relationship between the two. All three form factors achieved scores better than 70.14, which, coupled with the SUS results presented in section 5.5.4, confirms H1: participants who took part in the study considered HARATIO to achieve acceptable usability on smartphone, mini tablet, and tablet form factors.

In addition to producing an overall score, the structure of the HARUS can be decomposed into two factors that can then be evaluated independently to determine scores for ‘manipulability’ and ‘comprehensibility’. Manipulability provides a measurement of ergonomics while comprehensibility provides a measurement of perception. High manipulability scores indicate the target device is easy to manipulate while high comprehensibility scores indicate the AR system—in this case, HARATIO—presents information in a format that is easy for users to understand. Scores for both factors are calculated in the same manner as the overall HARUS by separating the responses into two groups of eight questions: 1–8 for manipulability and 9–16 for comprehensibility.

The graphs in Figure 5.14 show the manipulability mean decreasing as form factor size increases, particularly with the tablet; this correlates with the observations discussed

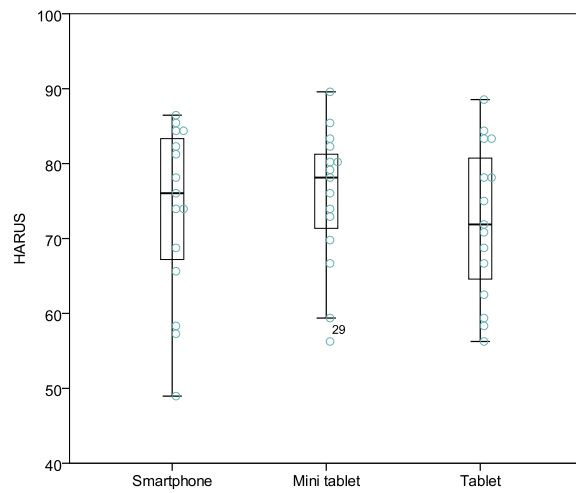


Figure 5.13: Participant HARUS responses for smartphone, mini tablet, and tablet devices

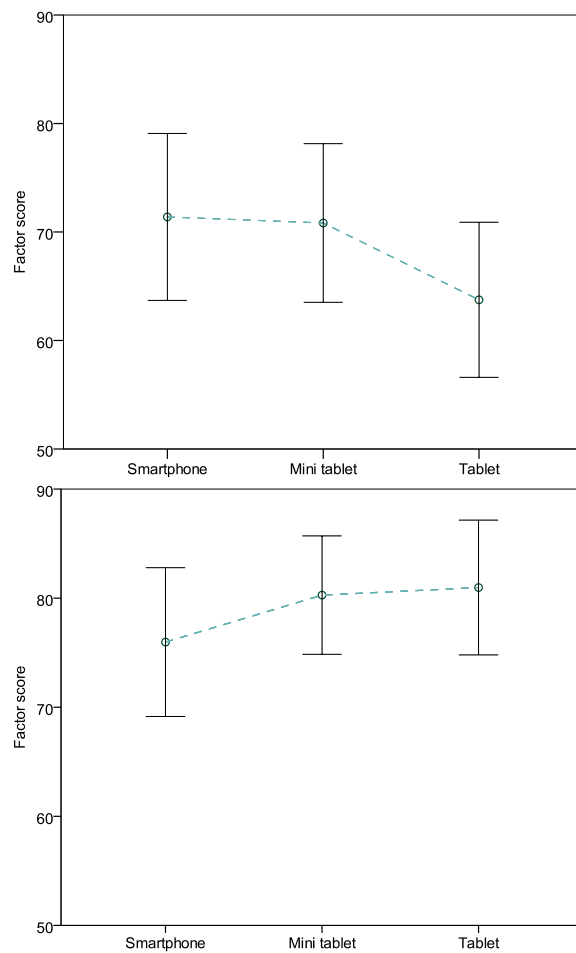


Figure 5.14: HARUS factor scores for 'manipulability' (top) and 'comprehensibility' (bottom) by device form factor.

in Chapter 3 where larger, heavier devices were more difficult to control. Contrastingly, means for comprehensibility rise as form factor size increases. As the primary benefit of using a larger device is the increased screen size, interface spacing and hit targets are both increased while the likelihood of screen occlusion is decreased. For both factors, there is an inflection point after which the corresponding factor score declines. For manipulability scores, it is the decline from mini tablet to tablet; for comprehensibility scores, it is the decline from mini tablet to smartphone. The factor analysis indicates the mini tablet form factor provides the best balance between comfortable device manipulation and interaction with HARATIO, which corresponds to its overall HARUS score bettering the other form factors. This agrees with the findings reported in Chapter 3, as well as this individual task SEQ results discussed in section 5.5.3, and further reaffirms the effect of screen occlusion (comprehensibility) on smaller devices and weight (manipulability) on larger ones.

5.5.6 Post-experiment Interview

After all tasks had been completed with all devices, participants were asked eight questions in a post-experiment interview. The purpose of the interview was to provide participants the opportunity to freely comment on the use of HARATIO and any difficulties encountered. Answers would provide insight into aspects of HARATIO's design that worked well and aspects that required further improvement. The questions covered the following topics:

- Preferred device form factor for authoring
- Issues encountered while using HARATIO to complete the tasks
- Use of the behaviour scripting editor for defining object interactivity
- Response to using a handheld and a tool like HARATIO to author AR content in situ
- Hardware factors considered important for mobile AR authoring

Preferred device form factor for authoring Just under half (47%) the participants cited the PC as their preferred device for reasons of familiarity and confidence with mouse interaction. The PC is still a dominant platform for content creation and can offer levels of input precision currently unmatched by touch screen devices and touch interaction. The response supports the assumption discussed in section 5.5.3 and also agrees with the conclusions of earlier work (Sambrooks and Wilkinson, 2013), carried out prior to this research, comparing the performance of mouse, gesture, and touch interaction using Fitts' law.

HARATIO was initially conceived as a mobile-only authoring tool. PC support was added to aid with the comparison of authoring collocated with the target environment (in situ) verses authoring in a separate location characterising a typical office setup. An interesting notion emerging from participants' comments regarding the PC was whether a tool like HARATIO could benefit from not being limited to handheld devices only. Exploring this idea further using the scenario described in Chapter 1, a class teacher wishing to create AR content for a field trip could scout the location themselves and begin authoring using their smartphone or (mini) tablet. In this context, these are the form factors that are most convenient and well-equipped for initial scene design. They afford quick placement of objects as well as rapid evaluation of content and interactions while the user remains in situ. However, once initial work has been completed, the teacher may wish to return

to his or her office and continue editing the scene using a PC to add finer details that benefit the use of a larger screen and more precise input device. As demonstrated in task two, the fact HARATIO's interface remained consistent between platforms enabled users to move between them seamlessly without needing to learn a separate interface. The only difference was the interaction mechanics that corresponded to the hardware being used.

In terms of the handheld devices, participants preferred the mini tablet (27%) over the tablet (20%) and smartphone (7%) for reasons similar to those discussed in Chapter 3. Many participants commented that they thought the smartphone was too small to comfortably display and interact with the interface. While the radial menu could be moved, it still occupied a large portion of the screen. When coupled with the display of the behaviour editor, this resulted in the amount of space left to display scene content reduced dramatically—almost to a quarter of the available screen real estate. Conversely, the tablet was considered large, heavy, and awkward to hold (somewhat slow to use, too) although a few did laud it for its bigger screen and ease with which on screen content could be viewed. The mini tablet seemed to strike the right balance between the two, as one participant (P4) succinctly summarised:

“... it had good screen real estate and I didn't have to compromise the way I held the device to complete the tasks.”

The similar SUS scores achieved by all three handhelds (section 5.5.4) suggests participants considered preference subjectively and not interrelated with usability. Though the mini tablet was the subjectively preferred form factor, the location of the rear camera in the top left corner of the device used was noted as being a minor annoyance for some while attempting to adopt a comfortable and secure grip. As most opted to hold the device in a landscape orientation, there were a few occurrences where it appeared that HARATIO had crashed—the screen would suddenly turn black—while attempting to track a marker. Further inspection quickly revealed that one of the participants' fingers was inadvertently covering the rear camera. While the placement of cameras on physical devices is beyond the control of this research, it does highlight the usefulness of a freeze technique that allows users to adjust their grip of the device and continue working on a scene without fear of interrupting tracking by inadvertently masking the camera.

Issues encountered while using HARATIO to complete the tasks When asked about the occurrence of any issues with HARATIO that affected task completion, all participants recalled at least one. The most common issue described was the perceived difficulty with translating one object on top of another. As discussed in Chapter 4, objects were translated relative to the point of selection. This was intended to allow objects to be positioned 'off centre', which would be useful in constructing compound models such as the archway in task two. The grid cell that would be used as the destination for the translation was determined based on a ray cast originating from the selection point and moving in the direction dictated by the authoring viewpoint (equal to the user's viewpoint of the scene) through the object to a grid cell. If the viewpoint was not orthogonal with the axes of the grid, and the selection occurred on the object's top or side face, the destination cell could be incorrectly interpreted as a cell adjacent to the base of the object. This would result in the translation operation being offset, positioning the object one cell greater or less than

expected. Users could easily become confused and frustrated by this behaviour as it would only occur in the described example. Ultimately, for those reporting difficulties with object translation, this is what happened and their comments were well-founded. The translation feature requires modification to eliminate this behaviour.

Comments were also made with regards to ambiguities in the use of certain operations. Participants noted that they were unsure as to when objects could be selected or deselected and there was a general lack of visual feedback to assist them. Some said they resorted to trial and error, moving between different menus until object selection could be changed. This mostly occurred when working on task two (scenario) where the need to edit multiple objects was a more frequent occurrence.

As discussed in section 4.4.2, HARATIO's selection model is object- rather than operation-centric. Any active activity operation that can affect a change in the selected object's state temporarily disables further selection changes. Given this behaviour, users expecting to activate the translation operation and then translate several objects consecutively would be confused as to why this was not possible. As there was no visual feedback to notify them that selection changes were disallowed when specific operations were activated, a trial and error approach was likely a natural response. To add to the ambiguity, when selection changes were permitted and the user wished to deselect an object, they would have to select another object or reselect the already-selected object. Analysis of video footage revealed many participants initially tried tapping on empty grid cells to deselect objects; however, this had no effect.

An object-centric design is still considered to have advantages over an operation-centric one in terms of simplicity and the ability to clearly communicate the target object for edits. If edit and behaviour activities apply specifically to the object that has been selected, it is clear where changes will be performed. The same level of certainty cannot be said for the alternative. For the translation feature, which uses a direct interaction for object movement, it would be conceivable for the user to inadvertently select a different object while trying to perform a translation, especially when multiple objects are in close proximity. Given the intrinsic inaccuracies involved with touch interaction, this is a usability concern.

In terms of the unnecessarily strict deselection process and lack of visual communication as to why selection changes were not permitted in certain circumstances, these are two areas that can be improved. Future iterations of HARATIO would implement solutions for both.

Other participant comments related to the assumption that scaling and rotation could be performed directly with a gesture, like the way images are rotated and scaled within many mobile photo applications. This is a valid assumption. Given translation is performed via a direct interaction gesture, it should not be surprising that there was an expectation that other RST operations would function similarly. The motivation for exposing rotate and scale operations indirectly via menus was primarily to alleviate occlusion issues. Performing multi-finger gestures for rotate and scale could easily obscure large parts of the screen making it difficult to accurately visualise the resulting changes. This would be especially noticeable on smaller screens. Translation could not occur indirectly given the nature of the operation, but it only requires a single finger interaction and is assisted via the callout that appears above the finger's selection point.

Finally, those participants who attempted to incorporate the use of the occlusion colour

into their task two scenes remarked that it was extremely difficult to reference where such objects were while constructing the scene, especially when building the archway. Objects assigned the colour were rendered within the author mode environment as they would appear in AR mode; that is, functioning as a depth mask and culling display of all intersecting virtual objects located behind them from any given viewpoint. In AR mode, this would reveal only the background corresponding to the feed from the device camera, but in author mode, it could effectively be a solid colour depending on the user's viewpoint configuration. Consequently, the size and location of these objects within the scene could only be reliably deduced when selected (due to the selection stroke). As discussed in section 6.1.2 of the following chapter, the author mode rendering of the occlusion colour was modified to address this issue.

Use of the behaviour scripting editor for defining object interactivity Overall impressions of the behaviour editor were positive with most participants (86%) stating they found it understandable and easy to use. The combination of words and symbols appears to have been helpful in clarifying behaviour script intent and the order in which triggers and actions occur. The drag-and-drop method of creating scripts was also considered intuitive. One participant did express concern over the discoverability of creating new behaviours, however, saying they were unclear as to which button on the menu to use. Elaborating, they recalled searching through various menus hunting for the right option. In the version tested, HARATIO presented two buttons on the main menu for cycling between behaviours scripts, labelled *Previous* and *Next*. The radial menu core indicated the loaded behaviour as its ordinal sequence out of the maximum number of scripts that could be attached to the object (e. g. 1/5). The design was intended to relieve users from having to consciously think about adding new behaviours by only presenting options to cycle through them as needed; the act of instantiating and attaching a new behaviour script to the object would be taken care of by HARATIO behind the scenes. Evidently, this approach caused some confusion and made it appear that there were multiple behaviours attached to every object even when the object had just been created. When looking to add a new behaviour script, it is no surprise the participant searched for an *Add* button rather than the *Next* button. As previously described in section 4.4.5.4, the representation of behaviour script sequence numbers within the radial menu core was amended to clarify the number of scripts attached to the object as well as the process by which new scripts could be added. Further discussion of this change also appears in section 6.1.3 of the following chapter.

Although behaviour script translation was seldom used, those that did (53%) thought it was useful in clarifying scripts as they became more complicated. Many even made a point to remark that they thought it would be especially helpful to users who were not programmers.

Unsurprisingly, participants who indicated previous programming experience expressed a level of inefficiency with the editor and said they would have preferred a code oriented (text-based) layout and the ability to view more than one script simultaneously. While these are valid comments, the design was intended to cater to inexperienced users first and would therefore naturally frustrate those who prefer to express their instructions in a more direct manner. One of these participants did, however, echo previous remarks

and acknowledge that the current implementation would be good for those without such experience.

Response to using a handheld device to author AR content in situ Although the experiment only simulated the differences between remaining in situ versus moving between separate locations, most participants responded positively to the notion of in situ authoring with a mobile device and considered there would be a benefit to remaining collated with the target environment. The ability to work more efficiently by evaluating content in real time as well as visualising the placement of virtual objects with respect to physical ones were all reasons frequently given.

A similar question asking whether they would consider using a mobile AR authoring tool in place of a PC-based one received a comparable response. All but one of the participants indicated yes—one even said they would do so for entertainment regardless of whether they had a specific task to complete. Other respondents placed a caveat on their answer: some stated they would only consider the larger devices as reasonable options while another said device speed would be an important consideration. Additionally, a few participants remarked that they thought a mobile AR authoring tool would be good for mock-ups with one elaborating on this by saying they would be happy to use it for simple scenes but would prefer a PC for more complex authoring tasks due to the demonstrably larger screen and more precise input.

Hardware factors considered important for mobile AR authoring The final interview question asked participants to rate various handheld device factors in terms of their perceived importance for mobile AR authoring. Eight factors were rated using a five-point Likert scale where one represented ‘extremely unimportant’ and five ‘extremely important’. Participants were instructed not to think too much about their responses, which would ideally result in ratings that would be representative of their recent experiences using HARATIO. A similar question was posed during the experiment discussed in Chapter 3 with the participants’ role being the key point of difference: in that experiment, participants evaluated existing task content and provided ratings from an end user perspective; in this experiment, the tasks were the creation of content and ratings were provided from an authoring perspective.

A summary of the responses is provided in Figure 5.15. Based on the data, screen size and input accuracy were considered the two most important factors. Given participant comments, this presents no real surprise. Smaller screens made it difficult to comfortably display interface elements and perform interactions without obscuring large portions of the screen. As touch input is inherently inaccurate, precise interactions would have been most difficult on the smartphone and further highlighted the screen size issue. One of the participants specifically mentioned experiencing difficulties with accurately selecting objects and menu items with this form factor. These issues would have likely been embellished by the fact participants were able to use HARATIO with a mouse on a large 27-inch PC display.

The least important factor was ergonomics. This is contrary to the results obtained previously where ergonomics was considered *most* important (refer to section 3.6.4). The inclusion of the freezing technique in HARATIO is surmised to be one of key contributors

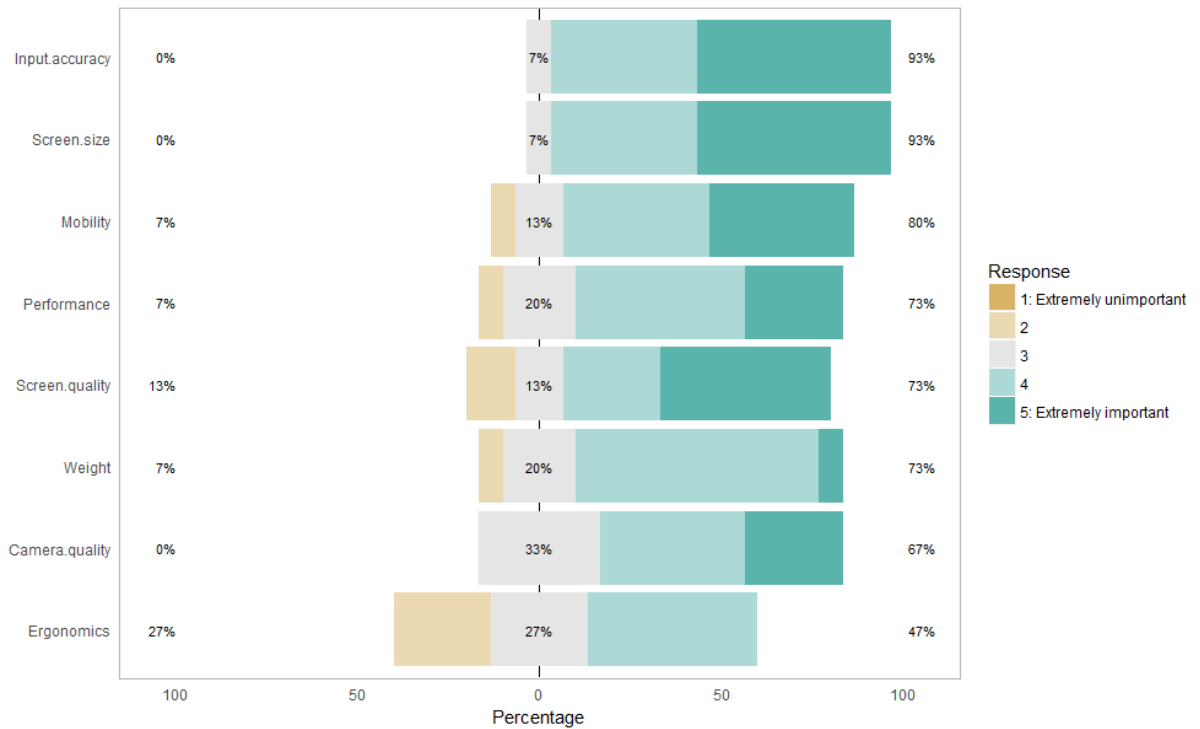


Figure 5.15: Distribution of participant responses to hardware factors considered important for mobile AR authoring.

for such a dramatic change. The AR application described in Chapter 3 did not include any form of freezing capability. For tasks requiring prolonged interaction, heavier devices placed uncomfortable, often fatiguing, strain on the participants’ hands, wrists, and forearms as they attempted to maintain a level of device stability. Authoring AR content will often require prolonged interaction, but as HARATIO includes the ability to freeze and stabilise the interaction space, participants were able to reposition the device to a more desirable pose and alleviate any uncomfortableness—a smartphone could be cradled in a hand while a tablet could be placed in a lap or on a desk. By removing the need for the device to be held in place for the duration of the task, participants likely considered ergonomics a non-issue and therefore rated it accordingly. In support of this notion, weight was the next lowest ranked factor.

During the interview, five of the participants did mention they experienced minor levels of discomfort while using the tablet, however, most conceded this occurred towards the end of the test and was likely a compound effect from using all three devices.

5.6 Summary

This chapter has evaluated the overall user experience of HARATIO as a mobile AR authoring solution suitable for different handheld form factors. Participants completed two tasks using three handheld devices and a PC. The PC was included as a reference device in order to assess participants’ perception of mobile AR authoring occurring in situ, collocated with the target environment, versus a typical office setup located separately. The tasks asked participants to assume the author role and use HARATIO to create AR scene content

adhering to task requirements. The first task was structured and required an existing scene be replicated per a predefined set of criteria; a completed version of the scene was demonstrated beforehand. The second task was unstructured and asked that a suitable scene be created according to a scenario description that involved the use of physical objects in addition to virtual ones.

HARATIO was found to be usable on all three handheld form factors, achieving scores greater than a benchmark of 70.14 in both SUS and HARUS usability instruments. Tablet and mini tablet form factors scored highest in each, respectively, which is counter to the usability findings reported in Chapter 3. Supported by these results, the inclusion of the freeze feature within HARATIO was demonstrated to be effective at improving the usability of larger devices. In each task, the tablet either exceeded or was within 1% of the completion rate recorded for the smartphone, suggesting the discrepancy in size and weight between these form factors was not an overwhelming factor as it was in Chapter 3. Consequently, it was expected that larger screens would result in more efficient task completion; however, this was not the case. The mini tablet recorded the lowest aggregate time for task one while the smartphone recorded the lowest aggregate time for task two. Further analysis suggested efficiency was influenced more by task familiarity than form factor.

Overall, the mini tablet was the most subjectively preferred form factor and proved easiest for authoring task content with HARATIO based on post-task SEQ data. It was also discovered to be the inflection point in HARUS manipulability and comprehensibility factors, which in turn correspond to overall ergonomics and perception of content. This correlates with the subjective preferences expressed in Chapter 3 and the mini tablet's balance between physical characteristics of adjacent smartphone and tablet form factors.

While the PC outperformed each handheld form factor in all measures; and was generally preferred due to familiarity, confidence, and precise input control; participant comments regarding authoring in situ were positive with all but one expressing a willingness to use a mobile AR authoring solution, such as HARATIO, in place of a PC-based one given an appropriate use case.

The following chapter outlines changes made to HARATIO in response to participant feedback and empirical data collected from the evaluation covered in this chapter. These changes relate to HARATIO in its final state as presented in Chapter 4. A further user evaluation is then described in which the radial menu and behaviour scripting editor interfaces were assessed in terms of their suitability for novice users.

HARATIO User Interface Evaluation *with Inexperienced Operators*

This chapter describes a second evaluation performed to assess HARATIO's radial menu and behaviour script editor interfaces and their suitability for novice users. These user interface components comprise much of the interaction experience users have with the tool while authoring AR scenes. The evaluation consisted of a formal user study and included participants inexperienced with HARATIO, programming, and AR to assess their reaction to respective interface items in terms of effectiveness, efficiency, and performance satisfaction—metrics used in the ISO Common Industry Format for Usability Test Reports. A subset of participants who had gained some experience with the tool from the previous evaluation (Chapter 5) was also included to provide a means of comparison and assessment of learnability effects.

The chapter begins by first describing iterative changes that were made to HARATIO in response to participant feedback and observations from the previous chapter. These changes exist within Chapter 4, which describes the final version of HARATIO, but are elaborated here in the context in which they were implemented between studies.

The remainder of the chapter then details the tasks and procedures used to evaluate HARATIO's radial menu and behaviour script editor interfaces. As described in Chapter 4, HARATIO's radial menu was designed to provide a coherent arrangement of operations via an organisational structure based on the use of distinct authoring activities. The menu adapts to user interactions accordingly, focussing attention on operations relevant to the current task. The behaviour script editor affords a method of script creation that does not presume programming experience and leverages the interplay between syntactic words and graphical symbols to form expressive conditional-like statements. Experimental tasks were devised to explore the user interface design of each respective component using existing AR scenes in erroneous or incomplete states. These scenes enabled participant attention to be directed to the interface being evaluated. The chapter concludes with results and corresponding discussion of observations.

6.1 Summary of HARATIO Changes

Prior to evaluating the radial menu and behaviour script editor interfaces, a series of modifications were made to HARATIO based on previous observations and participant feedback from the experiment presented in Chapter 5. These modifications addressed aspects of the interface the data suggested could be improved and are summarised below.

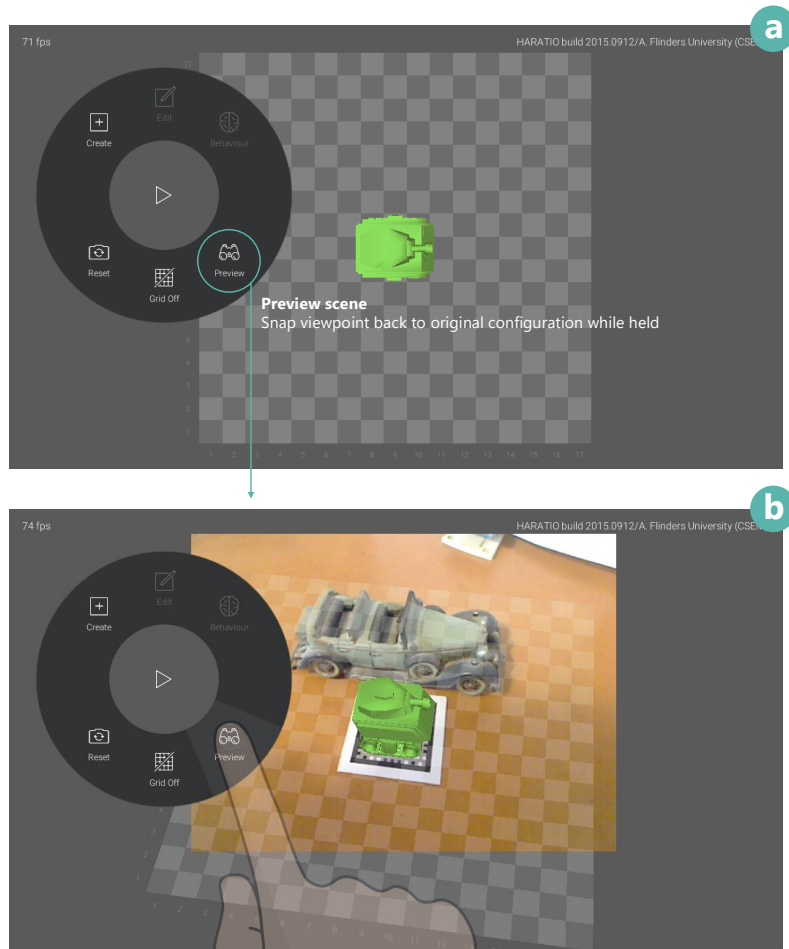


Figure 6.1: Earlier implementation of author mode viewpoint configuration. The default appearance resembled that of a typical 3D modelling tool (a). Holding down a **Preview** button enabled the scene to be viewed against the static background texture element (b).

6.1.1 Author Mode Viewpoint Interaction

Initial implementations of viewpoint interaction defaulted to an unlocked state when author mode was entered and concealed the background texture element. Content orientation was also adjusted to be orthogonal with the viewpoint, resulting in a default authoring appearance representative of a typical 3D modelling tool (Figure 6.1a).

To enable scene content to be referenced against the static background texture element, a **Preview** button on the root menu could be held down to snap the viewpoint back to its original configuration and reveal the texture (Figure 6.1b). The preview corresponded to the perspective of the scene when the transition to author mode was initialised (section 4.4.1). Releasing the button would snap the viewpoint back to its prior position and orientation, allowing authoring to continue. A **Reset** button returned the viewpoint to its default state equivalent to when author mode was first entered; it did not reveal the background texture. Finally, a **Grid** toggle control enabled the visibility of the scene grid to be toggled to aid with previewing the scene. These menu items are depicted in Figure 6.2a.

While post-experiment interview feedback discussed in section 5.5.6 did not reveal any

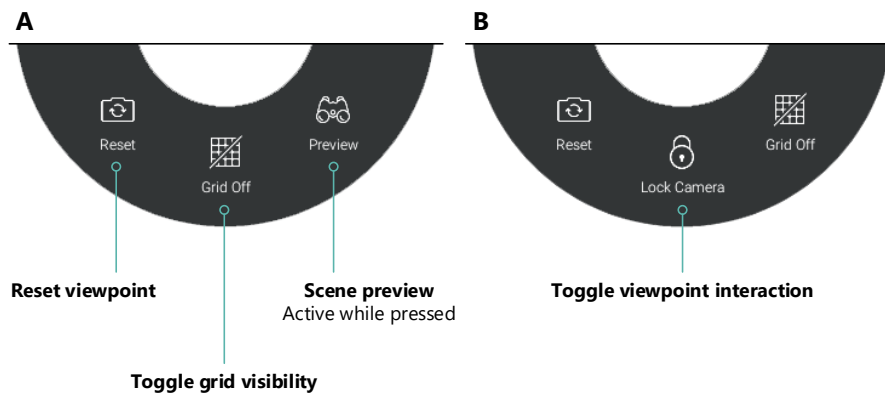


Figure 6.2: Original viewpoint control actions available in the lower section of the root menu level.

major issues with this approach, discussion with peers and experts raised concerns over concealing the real world reference provided by the background texture during authoring. While the scene could be previewed, as described, its provision as a secondary function that was mutually exclusive from authoring activities meant creating and editing virtual objects could not occur with a persistent reference to their intended location within the physical environment.

An amendment to this implementation restored the physical reference by fixing the viewpoint and not concealing the background texture element when author mode was activated. Providing an option to preview the scene became redundant as the new default configuration effectively always displayed a preview. Viewpoint interaction was demoted to a secondary function that could be toggled on or off based on each user’s authoring requirements. The goal was to adapt the default experience to be comparable to many existing mobile AR authoring tools, discussed in section 2.7, where the viewpoint of the scene when paused was the same as the viewpoint of the scene during authoring.

The bottom row of items on the root menu was modified to remove the *Preview* button and replace it with a viewpoint interaction toggle control (Figure 6.2b)—the label used the word *camera* instead of *viewpoint* as the former was considered less technical. *Reset* and *Grid* toggle menu items remained as before with *Reset* restoring the viewpoint to its default configuration by reversing all adjustments.

The changes discussed configure HARATIO to operate like many existing mobile AR authoring tools whereby the frozen camera image remains visible and the authoring viewpoint fixed. Adjusting the viewpoint can be achieved via an unfreeze-reposition-refreeze approach or, alternatively, users can choose to unlock the viewpoint to quickly gain control over their perspective of scene content. As discussed in section 4.4.1, this is beneficial when needing to access objects occluded by other objects. The revised implementation ultimately provides authoring flexibility and equally caters to novice and experienced users alike.

6.1.2 Occlusion Colour Appearance Within Author Mode

In response to participant feedback, the author mode representation of occlusion-coloured objects was modified to make it easier to identify them and discern their location within a scene. Originally, proxy objects assigned the colour were displayed in AR and author

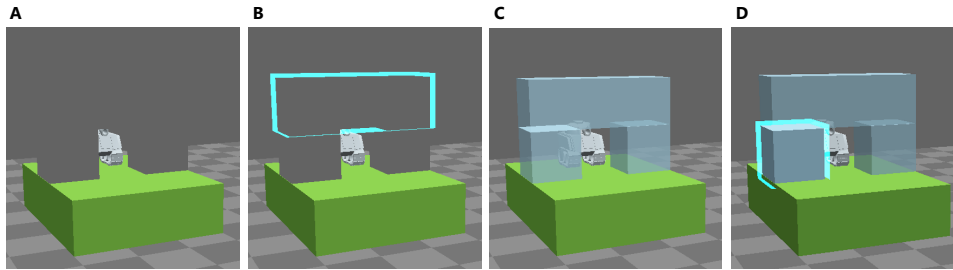


Figure 6.3: Original depiction of occlusion-coloured objects within author mode (a–b); such objects were difficult to visualise due to their appearance representing that of a live AR view. Modifications improved visibility of the occlusion colour within the authoring editor but still made it distinguishable from standard colours (c–d).

modes using the same depth mask material. As participant remarks revealed, this made them difficult to visualise during authoring as their appearance provided no visual cues. Consequently, they frequently appeared invisible, merging into the background. The example provided in Figure 6.3a demonstrates this issue. In the screenshot, it appears as though only two occlusion-coloured objects exist: the columns either side of the tank. This is inferred by the grey tank, green platform, and grid appearing partially masked. Figure 6.3b reveals that a third crossbeam is also located in the scene atop the columns, but without being selected, its placement within the scene is unclear.

In response, modifications were made to the author mode appearance of the occlusion colour to clear up ambiguities involving the number and location of associated proxy objects. Rather than render such object using the depth mask method, a semi-transparent preview material was added as a substitute offering much improved visibility (Figure 6.3c). The use a semi-transparent material in place of an opaque one provides a hint that the objects are intended to function as proxies and not exist as standalone virtual content. It also allows users to easily visualise which parts of other objects will be obscured. The transparency is temporarily removed whenever the object is selected, as depicted in Figure 6.3d, to clarify selection and indicate object state without visual distractions from nearby objects partially visible behind.

When the user leaves author mode and returns to AR mode, proxy objects assigned the occlusion colour are rendered as per the original depth mask implementation described in section 4.4.4.1. If the user re-enters author mode at a later date, their appearance reverts back to using the semi-transparent preview material.

6.1.3 Behaviour Activity Menu

Feedback on the use of the behaviour script editor highlighted confusion over how new behaviour scripts could be created. Originally, the core of the Behaviour activity menu indicated the loaded behaviour script in terms of its sequence number along with the *maximum* number of scripts that could be assigned to the object, irrespective of how many the user had actually programmed. As users cycled to the next behaviour script using the associated button, HARATIO would dynamically create a new script for the indicated sequence number in the background if one didn't already exist. Given the maximum number of scripts that could be attached to an object was set at five, this approach unintentionally

gave the appearance that all virtual objects contained five scripts, including those that had just been added to the scene. Some participants' natural reaction was to therefore search for an 'add' button to create a sixth script.

This design was modified to clear up this confusion by showing only the number of behaviour scripts currently attached to an object rather than the maximum that could be attached (refer to Figure 4.24 in Chapter 4). The Previous and Next buttons still allowed users to cycle between behaviours as before, however, the Next button was amended to include an (Add) suffix whenever the loaded behaviour was the last in the current sequence and the maximum had not been reached. Selecting the annotated Next button in this state would explicitly add a new behaviour script to the object and advance the total number in the sequence.

6.1.4 Object Selection

Post-experiment interview responses discussed in Chapter 5 noted a level of uncertainty among participants regarding when individual objects could be selected or deselected. As discussed in section 4.4.2, HARATIO adopts an object-centric selection model in which selection is temporarily disabled while operations that can modify an object's properties are active. To better communicate when this behaviour interferes with a selection attempt, the notification system discussed in subsection 6.1.5 was implemented, allowing a message to be displayed to the user alerting them to the conflict and offering a solution to resolve it.

The way objects are deselected was also modified to better reflect the way many participants were observed attempting deselection. Rather than require a different object be selected or the current object reselected, the deselection routine was altered to additionally recognise a deselection as any interaction with an unoccupied grid cell.

6.1.5 Visual Feedback Improvements: Notification System

Participant confusion regarding object selection restrictions with edit and Behaviour activity operations highlighted a need to improve the level of visual feedback provided. To help resolve selection confusion and generally improve feedback across all aspects of HARATIO, a notification system was implemented to enable timely communication with the user, both in terms of providing explanation on operation status and offering helpful hints and suggestions pertinent to the current application configuration.

Notifications slide in from the top of the screen when displayed and reciprocally slide out when dismissed. They are intended to be unobtrusive and not encroach on the user's view of the scene or interface. While pop-up dialogs can provide the same level of information, they are often modal and are disruptive by appearing centrally within the interface. Rather than disappearing automatically, dismissing a dialog typically involves explicit interaction from the user.

HARATIO's notification implementation is analogous to the way notification messages operate on modern mobile platforms whereby, once visible, the message has a finite display time before dismissing itself. This behaviour has been extended to also support undefined durations that are 'sticky' and instead require a condition be satisfied before being dismissed; in this case, the message content will describe the condition. In both

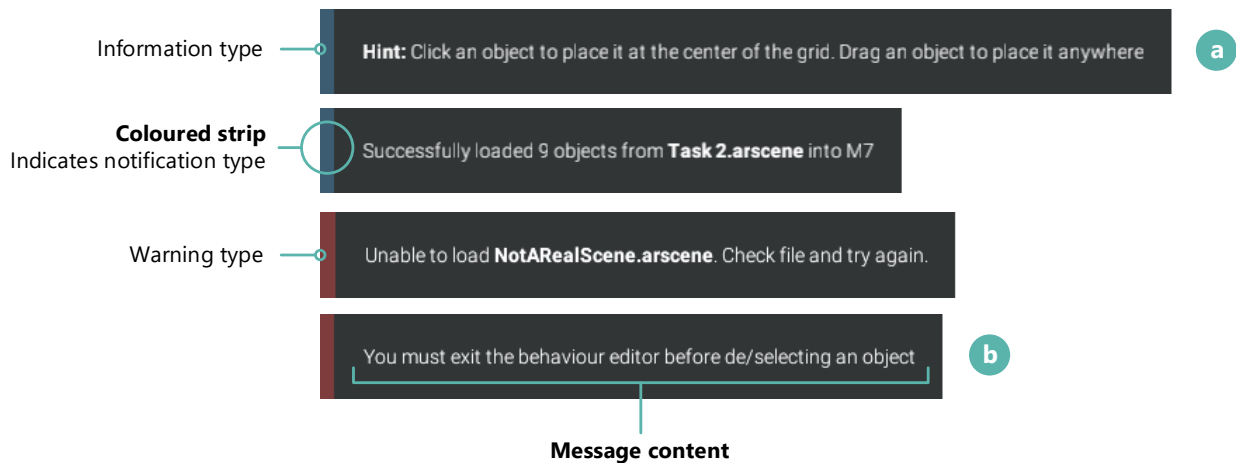


Figure 6.4: Notification messages comprise a coloured vertical stripe, denoting the *type*, and the message body. Two types of notifications are supported: *information* and *warning*.

cases, no direct interaction with the notification is required from the user.

The presentation of a notification is comprised of two components: a coloured vertical stripe along the left edge that identifies the notification type and the body section that contains the message content (Figure 6.4). As with the radial menu’s use of colour in distinguishing between authoring activities, expressing notification types with a coloured stripe provides a visual cue as to their importance.

Two types are currently available denoting *information* and *warning* level notifications. Information notifications comprise messages regarding status updates or hints. They are displayed with a light blue stripe. Hint messages are intended to provide just-in-time assistance on the discoverability and use of certain HARATIO operations. The example illustrated in Figure 6.4a shows a notification containing a hint that describes how to add new virtual objects to a scene from the object library panel—either via single-tapping or dragging as per the description in section 4.4.3.2. This notification is configured to appear whenever a user first accesses the Create activity. Given the instructional intent of the notification message, it is configured to be ‘sticky’ and remains visible until dismissed by way of the user creating an object or exiting the activity. If the user exits the activity without creating any objects, the notification will reappear when the Create activity is next accessed. It is only by creating an object that the notification will be permanently dismissed as a confirmation the user has understood its content.

Warning notifications communicate errors that may have occurred or provide explanations for why particular operations cannot be performed. They are displayed with a red stripe. An example of a warning notification is an alert displayed whenever an activity imposes a restriction that prevents object selection from being modified. Figure 6.4b shows an example that would appear if the user attempted to modify object selection while the Behaviour activity was active. This addresses participant feedback discussed in section 5.5.6 regarding confusion with HARATIO’s object-centric selection model, which was one of the motivations for improving the level of visual feedback provided.

To avoid overcrowding the interface and overburdening the user, successive notifications do not appear concurrently but rather consecutively. Each notification is processed by an internal queue and is displayed as soon as the previous notification has been dismissed.

If the queue is empty, the notification is shown immediately.

6.1.6 Scene Serialisation

In preparation for the evaluation discussed in later sections of this chapter, scene serialisation was incorporated into HARATIO to facilitate saving and loading AR scenes. Serialisation of a scene includes all virtual objects and all associated behaviour scripts. Objects are saved with all properties intact; this includes colour, scale, position, rotation, and internal name (a unique reference identifier). Scene files are stored in a binary format and are designated the file extension `arscene`. To safeguard against write errors when an existing scene file is to be overwritten, a copy of the existing scene is created with the extension `artemp` before any new bytes are written. If an error is encountered while writing the new scene file, the old scene will be automatically restored.

The serialisation process occurs independently of fiducial markers. This allows saved scene files to be loaded for any marker, not just the one that was used when the scene was originally created. Furthermore, should the tracking solution be modified or updated in a future iteration, serialised scenes would continue to be usable. Ultimately, scene serialisation is important for a mobile AR authoring tool as created content will need to be easily distributable between devices and to other interested users.

Schmalstieg and colleagues' (2011) AR 2.0 concept lists content distribution as a key area in addition to the provision of easy to use authoring tools. Considering the example use case described in Chapter 1, an AR 'work scene'—an AR version of a worksheet—could be authored by a teacher using their handheld device and then uploaded to a central school server where it could subsequently be distributed to student devices at the relevant time (or place). Students would then be able to concurrently participate in a class exercise using any arbitrary fiducial marker. Improvements to the tracking implementation that absolves the need for markers would provide even more flexibility.

From an experimental perspective, serialisation allows multiple participants to efficiently evaluate identical scenes on similar or different devices, permitting focused assessment of specific user interface components or application features. This approach was used in the experiment discussed in this chapter. Access to scene serialisation functions is discussed in the following subsection.

6.1.7 Slide Over Menu

Access to scene serialisation required the creation of additional menu options. Rather than include such options in the radial menu, which was focused on exposing operations related to the three AR authoring activities, an additional menu interface was created to serve as the central place for subsidiary application features. The menu functioned in a slide over capacity and appeared modally from the left edge of the screen when activated; when not active, it remained hidden and consumed no additional screen space. An illustration of the menu is provided in Figure 6.6.

A `File` button was added to the status bar displayed at the top of the screen to facilitate revealing the menu. The iconography of the button consisted of three stacked horizontal lines suggesting a list-based menu structure. Although this iconography is colloquially referred to as the 'hamburger' icon, its roots can be traced back to the original Xerox Star

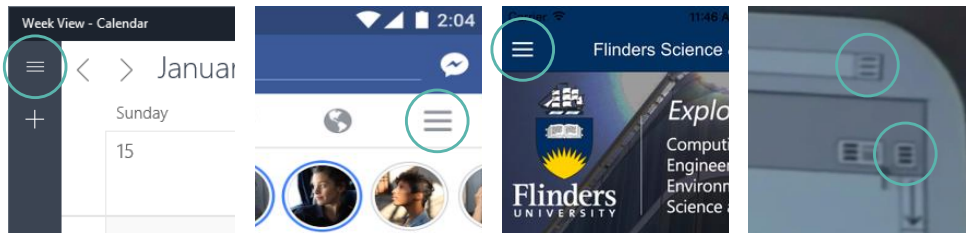


Figure 6.5: Examples of hamburger button use in old and modern interfaces. From left to right: Windows 10 Calendar, Facebook (Android), Flinders University Science and Engineering Quiz (iOS), and the Xerox Star (ca. 1981).

interface. Many mobile and desktop applications now use this style of button to denote menu hierarchies that are revealed upon its selection (Figure 6.5), so its use and affordance within HARATIO should be familiar, especially to users of modern mobile devices. In their respective developer documentation, both Google and Microsoft advocate using it for toggling collapsible menus in screen-constrained contexts.

Once the File button has been selected to reveal the menu, it is replaced with a back button that enables the menu to be dismissed. This button adopts the same iconography as the back button used in the radial menu core to navigate back through the menu hierarchy and therefore provides similar affordances (see section 4.2.2). Selecting an item in the menu will also dismiss it unless further action is required from the user.

The slide over menu was used extensively throughout the user evaluation of HARATIO described in this chapter and was populated with three items. The first two items comprised scene serialisation operations to enable AR scenes to be saved and opened (loaded), respectively. Participants were required to use the open operation when beginning a task to load the required task scene file, and the test moderator used the save operation after the task was finished to serialise the participant’s attempt.

Selecting the Save or Open item causes the contents of the menu to be replaced with a simplified file dialog containing a list of AR scene files stored on the local device. The vertical nature of the menu is well suited to displaying a scrollable list that may grow over time to overflow the available screen space. Opening (loading) an existing scene involves selecting the associated file and tapping the Load scene button. HARATIO will de-serialise the file contents and replace all objects in the current scene with those contained within the file. Once all objects have been successfully loaded, a notification is displayed to the user using the notification system discussed in section 6.1.5. If an issue is encountered during the operation, a different notification is displayed. Both notifications are illustrated in Figure 6.4.

Saving a scene operates similarly. Instead of selecting an existing scene, the desired file name is entered into the available text field using the on-screen keyboard of the host device. Tapping the Save scene button proceeds to serialise the scene content into a new arscene file using the file name specified (section 6.1.6). File extensions will be added automatically; however, if an extension is specified along with the filename, it will not be duplicated. Alternatively, an existing scene may be overwritten by selecting it from the list of those currently stored on the device. In this case, the existing file name is simply copied to the text field, and the save process continues as before. As with opening a scene, a notification is displayed once the save process has been completed. Serialisation does

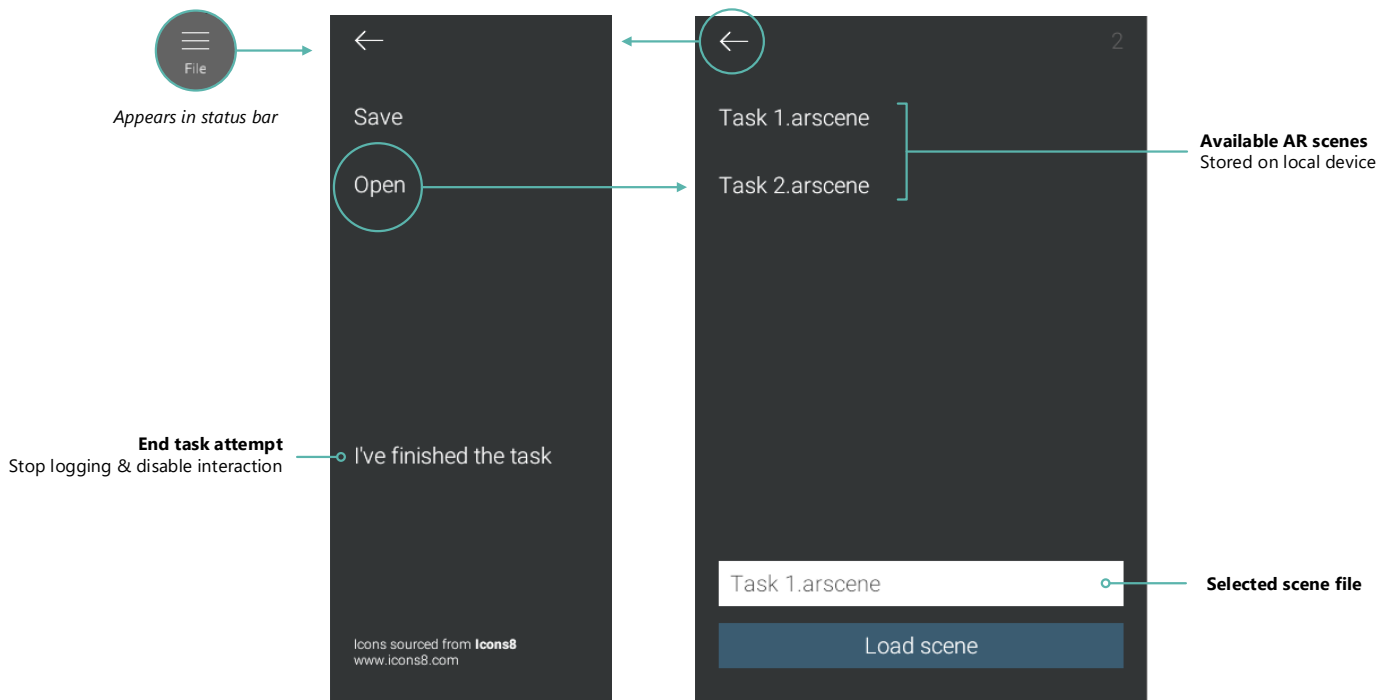


Figure 6.6: Overview of slide over menu interface

not process scenes containing no objects, so in this case a different notification would be displayed.

The last item contained within the slide over menu was used by participants to indicate when they had finished a task attempt. The item, labelled *I've finished the task*, would record the task end timestamp, stop internal logging, temporarily suspend further interaction with HARATIO, and present a notification to the participant instructing them to hand the device back to the test moderator. The item was intentionally separated from save and open operations to prevent it being accidentally selected during initial task procedures. The save item remained accessible to enable the task attempt to be serialised for later analysis.

6.2 Tasks

The two user interface components evaluated in this chapter, the radial menu and behaviour script editor, comprise key aspects of HARATIO's user experience. Users are likely to interact with both frequently while authoring typical AR scenes. Tasks were therefore designed with a singular focus so that participants could concentrate on one aspect of the interface at a time. Each task included pre-authored elements that were saved within AR scene files ready for distribution to participants during testing.

6.2.1 Task 1: Radial Menu

The first task explored the radial menu interface. Participants were presented with a pre-authored scene that was intended to spell out the abbreviation AR using a series of

cube objects. However, the initial state of the scene contained a series of errors: some of the cubes were in incorrect positions or orientations, some had incorrect colours assigned, and others weren't cubes at all but instead spheres or cylinders (Figure 6.7b). The goal was to correct the errors and produce a scene that looked like Figure 6.7c.

Completing the task would involve participants traversing the menu hierarchy of HARATIO and accessing appropriate activities and operations to correct the various scene errors. Depending on the approach taken, the operations used to achieve this could include the following: Rotate, Scale, Translate, Clone, Create, Delete, and Colour. To assist with the placement of objects in their correct locations, a grey silhouette depicting the outline of the letters, constructed out of plane objects, was included as part of the scene (Figure 6.7a). Participants could use this silhouette as a guide when determining the correct position and orientation of each object. Instructions regarding the goal of the task along with an image depicting the completed state were provided in an accompanying task reference sheet.

An example of one method that could be used to correct the errors is provided below. The example demonstrates how all of the operations could be used. Each step references the object labels visible in Figure 6.7b.

- Scale object 1 to correct length
- Move object 2 down two grid cells
- Rotate object 3 ninety degrees (left or right) and change its colour to red
- Delete object 4
- Clone object 1 and move it to the position previously occupied by object 4
- Change the colour of object 5 to red
- Move object 6 down two grid cells and rotate it ninety degrees (left or right)
- Move object 7 down one grid cell
- Delete object 8
- Create a new cube object in the position previously occupied by object 8
- Scale object 9 to correct length
- Change colour of object 11 to red

Alternate approaches could also be used that omit one or more of the operations. Rather than instruct participants on a specific method to complete the task, it was decided to let them design their own solution to gauge how intuitive and accessible the menu design was for someone unfamiliar with the concepts and nomenclature of 3D modelling. Successful

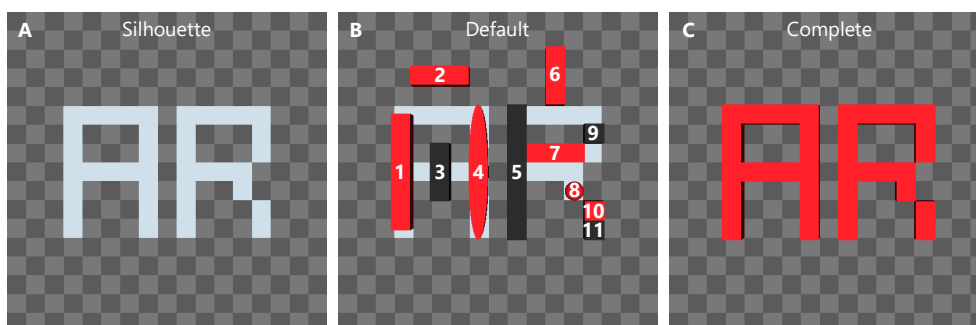


Figure 6.7: Task one scene showing: the silhouette outline of letters (a); the default state provided to participants (b); the completed state (c).

completion was therefore predicated on the final scene produced, not which particular operations were used in the process.

6.2.2 Task 2: Behaviour Script Editor

The second task explored how well the behaviour script editor enabled users without programming experience to define scripts using a mix of syntactic words and graphical symbols. Participants were again presented with a pre-authored scene that contained a series of virtual objects—they would not be required to create additional objects. The scene included a grey path, a red destination plate, a green archway, and a red tank (Figure 6.8). The path and destination plate were constructed out of plane objects and the archway out of cube objects. The archway was positioned over the path at approximately the half way point. The tank was initially positioned at the beginning of the path with two behaviour scripts attached, one of which was misconfigured and the other incomplete. The first behaviour was intended to change the tank's colour from red to green after a delay of four seconds. This would coincide with the tank passing under the archway when the second behaviour was correctly implemented. In the state provided to participants, however, the target colour was set to orange instead of green. The second behaviour contained an `Always` trigger but no actions. This behaviour was intended to move the tank along the path until it reached the destination plate at the end, after which it would stop moving and change colour back to red to match the plate. The accompanying task reference sheet provided a description of what both behaviours should accomplish and additionally included a pictorial overview of the completed scene in action at various points, illustrating the interaction between the tank, its behaviour scripts, and the environment. For a description of the Behaviour script editor along with the triggers and actions that comprise the language, refer to section 4.4.5.

Completing the task would require participants use HARATIO's behaviour script editor to correct both behaviours so the tank acted as intended. No additional tools or input devices were necessary to achieve this. As the first behaviour was functionally and syntactically valid, fixing it would simply involve the participant reconfiguring the `Colour` action to set the correct target colour of green. Correcting the second behaviour would be more involved as all necessary actions would need to be defined. As the tank was required to move along the path in a clockwise direction, a combination of `Move Forward` and `Turn Right` actions would be needed. Once the movement logic was in place, a final `Colour` action could be appended to reset the tank's colour to red, which would correspond to the tank reaching the red destination plate. As with task one in the experiment discussed in Chapter 5, movement could be specified stepwise or using repetition via the `Repeat` action. The choice was up to the participant. Example completed behaviour scripts utilising stepwise actions are illustrated in Table 6.1. Successful completion was based on whether the implemented behaviours were functionally and syntactically valid and produced the prescribed outcome.

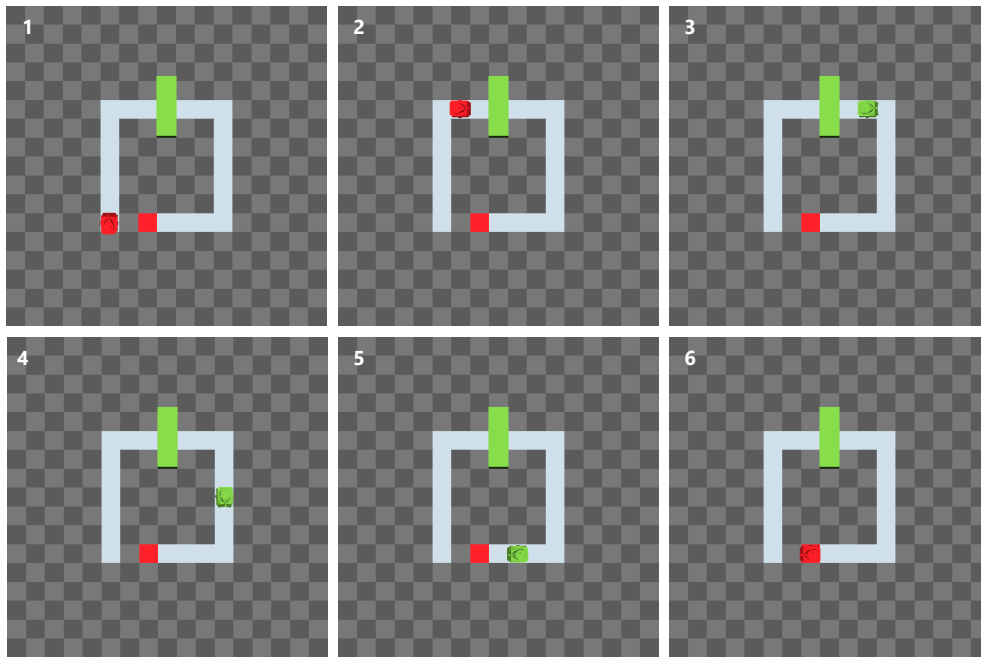


Figure 6.8: Task two scene showing: the default state (1); a progression of the tank moving along the path and changing colour (2–6).

6.3 Hardware

As the evaluation reported on in this chapter was concerned with key user interface components of HARATIO, testing involved the use of a single form factor only. Multiple mini tablet devices were available for testing, compared with only one smartphone and one tablet, so the mini tablet was selected as the form factor on which to conduct evaluations—this also enabled concurrent sessions to occur. In the previous evaluation discussed in Chapter 5, the mini tablet performed well in analysis of task performance satisfaction and handheld AR usability, and was subjectively preferred ahead of all other handhelds.

Device hardware and configuration remained as per the previous two experiments (Chapters 3 and 5) apart from an updated version of HARATIO that incorporated the changes summarised in section 6.1. Specific hardware details for the mini tablet device are available in section 3.3.

6.4 Procedure

Advertisement for the experiment sought volunteers inexperienced with programming and AR¹. Thirteen volunteers fitting these prerequisites offered to participate. Of the thirteen, seven were female and six were male. The age range was 18 to 75. One participant indicated English was not their native language. In addition, five participants who had previous experience with HARATIO from the study discussed in Chapter 5 were asked to volunteer their time once again. All five were male, aged between 21 and 40, current post-graduate students, and had at least some programming knowledge.

The experiment used a within-subjects design with each participant attempting both

¹ A modification to the ethics application pertaining to the experiment discussed in Chapter 5 was submitted prior to the commencement of this study. Appendix E contains a copy of the final approval notice.

Table 6.1: Stepwise behaviour (B) script examples for completing task two.

B#	Script	Trigger	Action
1	When [4 seconds have elapsed] do [set colour to green] end	Wait	Colour
2	When [the behaviour starts] do [move forward 6 grid cells] then [turn right 90°] then [move forward 6 grid cells] then [turn right 90°] then [move forward 6 grid cells] then [turn right 90°] then [move forward 4 grid cells] then [set colour to red] end	Now	Move Forward Turn Right Move Forward Turn Right Move Forward Turn Right Move Forward Colour

tasks; the order of the tasks did not change. Before beginning, participants were asked to complete a background questionnaire to collect demographic information on age range, gender, and current occupation. They were then given a brief introductory presentation by the test moderator. Content included a definition of AR for those unfamiliar with the term, capturing fiducial markers with HARATIO in preparation for authoring, basic concepts behind navigating around the interface, and the format for the remainder of the session. The presentation was delivered via a series of static slides that were narrated over by the test moderator. Descriptions were intentionally vague regarding the use of the radial menu and behaviour script editor as these were the focus of the study. However, the use of the slide over menu was covered in sufficient detail so participants were clear on how they could open AR scene files and indicate when they had completed a task. The introductory component lasted approximately five minutes and participants were encouraged to seek clarification on any aspect of the session format they did not understand.

Participants were then directed to a testing station. The testing station consisted of a table on which the required testing materials were located (Figure 6.9). These comprised a mini tablet device, a single fiducial marker to be used for both tasks, two task sheets, two post-task questionnaires, and a printed copy of the slides used in the introductory presentation. A copy of the task reference sheets and post-task questionnaire is provided in Appendix H. Task related documents were collated according to the sequence they would be required; that is, the task one reference sheet appeared before the task one questionnaire sheet, which in turn appeared before the task two sheet, and so forth.

In addition to the use of a head-mounted video camera, the experiment discussed in Chapter 5 also employed screen recording software to provide a level of redundancy and a clearer, less obstructed view of HARATIO in use. A similar screen recording approach was adopted for this study in lieu of a head-mounted setup. Even though modifications to HARATIO added support for saving AR scenes, capturing the authoring process in real time was still considered valuable as a means to evaluate the method in which participants constructed their solutions and highlight any notable differences between them.

Upon taking a seat at the testing station, the test moderator started the screen recording software, loaded HARATIO, and handed the device to the participant. They were then asked to read the task one reference sheet and begin the task when ready. The first line

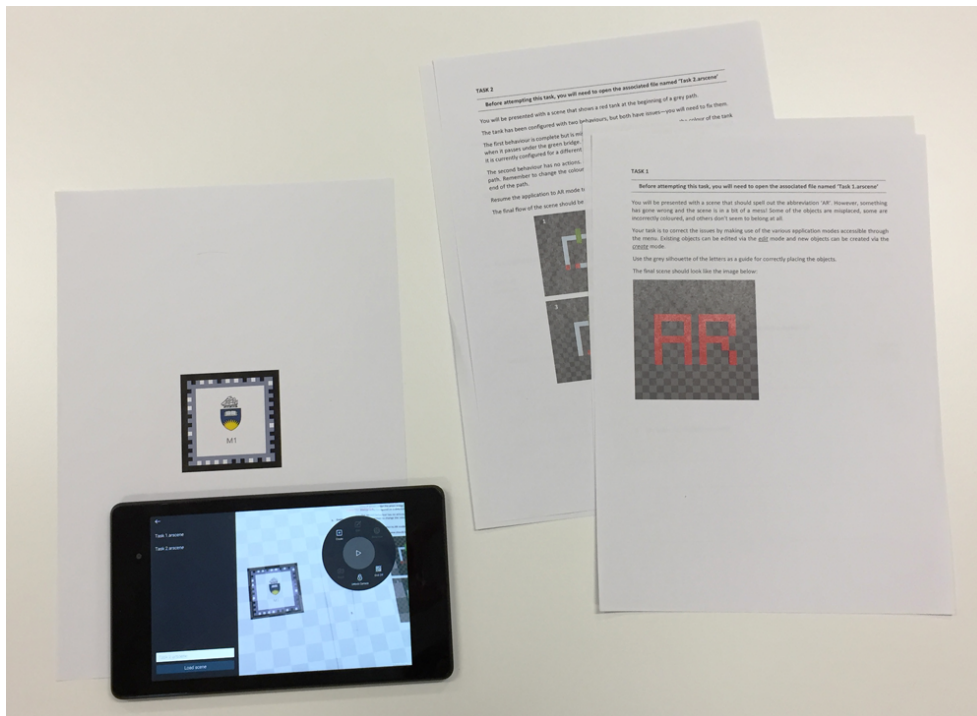


Figure 6.9: Testing station setup with materials provided to participants

of the reference sheet instructed participants to open the associated AR scene file via the slide over menu. Scene files were named according to the task they referred to (for example, *Task1 . arscene*) and contained pre-authored content necessary for completion. The radial menu evaluated in the task was initially located in the top-left corner of the device screen. As per discussion of the menu in section 4.2.2, however, this location could be changed as desired to overcome scene occlusion or better suit the participant's preferred hand. Completing a task attempt was indicated by selecting the appropriate option within the slide over menu (section 6.1.7). Once selected, a notification message would appear instructing the participant to hand the device back to the test moderator. The test moderator would then save the participant's work as a new AR scene file, named according to their anonymous identifier, and stop the screen recording.

The participant was then asked to complete a short post-task questionnaire. The questionnaire comprised three questions: a Single Ease Question (SEQ) (Sauro and Lewis, 2012, p.214) rating of how easy the task was to complete using the interface component being evaluated—the radial menu for task one and the behaviour script editor for task two—a subjective rating of how confident the participant was that they completed the task successfully, and space for any relevant feedback. Answers to the first two questions were given on a seven-point Likert scale with a rating of seven corresponding to the most positive response; that is, 'very easy' for question one and 'extremely confident' for question two.

Once the post-task questionnaire had been completed, the test moderator reset HA-RATIO and restarted the screen recording software in preparation for the next task. The device was then handed back to the participant and they were instructed to read the task two reference sheet and begin when ready. As with task one, the participant was required to open the relevant AR scene file prior to commencing their attempt and indicate

when they were finished using the option provided in the slide over menu. A post-task questionnaire was completed following the task. The session concluded once both tasks had been attempted.

6.5 Hypotheses

The user evaluation tested five hypotheses that explored HARATIO's radial menu and behaviour script editor interfaces in terms of effectiveness, efficiency, and performance satisfaction.

H1. Radial menu effectiveness does not benefit from experience. This can be decomposed into two testable sub hypotheses:

H1a. Participants without HARATIO experience achieve the same level of task success as participants with HARATIO experience.

H1b. Participants without HARATIO experience require no more assistance to use the radial menu than participants with HARATIO experience.

H2. Experience benefits interface efficiency. The percentage of a task completed per minute is higher among participants with HARATIO experience compared to those without.

H3. All novice participants without HARATIO or programming experience create syntactically valid behaviour scripts that result in no failed task attempts.

H4. Novice participants without HARATIO experience rate the performance satisfaction (SEQ) of the behaviour script editor greater than 4.

6.6 Results

This section presents results and observations obtained from the evaluation of the radial menu and behaviour script editor interfaces. Analysis of the data comprised assessment of three usability metrics derived from the Common Industry Format for Usability Test Reports [AS/NZS ISO 25062:2006 \(R2016\)](#): effectiveness, efficiency, and (performance) satisfaction. These metrics support the examination of the hypotheses listed in the previous section.

When examining the data, participant results were split into two groupings based on previous experience with HARATIO. Those that had not used the tool prior to the study, and additionally were inexperienced with programming and AR, comprised the *inexperienced operators* (IO) group. Those that had used HARATIO before comprised the *experienced operators* (EO) group. In total, there were thirteen IO users and five EO users. The two post-experiment groupings permitted a comparison between the effects of experience, as a substitute for formal training, in addition to usability.

6.6.1 Task 1: Radial Menu

Results from the first task were used to evaluate the radial menu in terms of providing effective and efficient access to create and Edit activity operations required for correcting

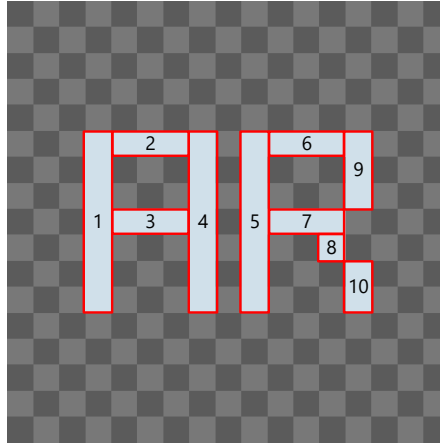


Figure 6.10: Zone distribution of the letters A and R. Each zone had to be precisely covered in order to be considered successful.

scene errors. Analysis compared the effects of HARATIO experience and included measures of task completion, task errors, menu assistance, menu efficiency, and participant satisfaction.

6.6.1.1 Effectiveness

Task completion was determined using a partial scoring method with reference to the graphic shown in Figure 6.10. The letters, A and R, were separated into ten zones. Based on the original state of the scene, discussed in section 5.1.1, each zone would require at least one edit operation to be completed—that is, be appropriately covered by one or more red cubes². Thus a participant could achieve anywhere between 0 and 10 points for their attempt. A score under 10 would indicate the presence of errors in the attempt, and any score of five or below was considered a failed attempt. The formula shown in equation 6.1 describes the calculation of aggregate completion rate (CR) percentage based on participant scores: n is the number of participants and r the result obtained (out of 10) by participant i for the task.

$$CR = \left(\frac{\sum_{i=1}^n r_i}{n} \right) \times 10 \quad (6.1)$$

Overall completion rate was 95%. Participants comprising the IO group achieved a completion rate of 95% and a mean score of 9.46 ($SD = 1.2$). By contrast, those comprising the EO group achieved a completion rate of 96% and a mean score of 9.6 ($SD = .89$).

While a binary task success rate is much simpler, it unreasonably dismisses attempts where only minor errors are made but the result is otherwise correct. Figure 6.12 shows examples of erroneous attempts from four different participants. In each case, only minor errors were made in 1 to 4 of the zones. Though the difference in completion rate between IO and EO participants was just 1%, H1a is nevertheless rejected: participants with HARATIO experience achieved higher task success compared to participants without.

Following the task, participants provided an indication of how confident they were

² Even though the term cube usually refers to a 3D object with six equally-sized faces, the term is hereinafter used to describe any object originating from a cube, regardless of size.

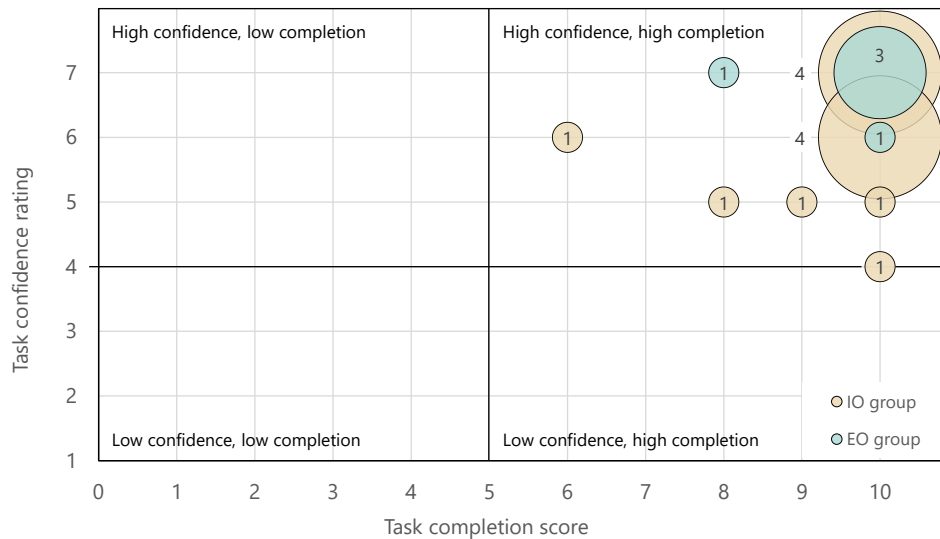


Figure 6.11: Relationship between task completion rate and perceived confidence level for task one. Numbers central or adjacent to bubbles indicate size (i. e. frequency).

that it had been completed successfully. This was given on a seven-point Likert scale where one represented ‘not at all confident’ and seven ‘extremely confident’. The overall mean confidence rating was 6.17 ($SD = .92$). Participants comprising the IO group recorded a mean confidence rating of 5.92 ($SD = .95$) while participants comprising the EO group recorded a mean confidence rating of 6.8 ($SD = 0.45$).

The graph in Figure 6.11 illustrates the relationship between task completion score and indicated confidence level. The plot is separated into quadrants that each represent one of four associations—the bottom right quadrant, for example, denotes task attempts that were successful but received low confidence ratings. According to Sauro (2011), a result where a user performs poorly in a task but indicates a high level of confidence that it is correct (the top-left quadrant) is called a ‘disaster’. The presence of disasters would indicate issues with the affordances provided by the menu design, leading to unnecessary or incorrect operations being performed, or otherwise signal the user’s misinterpretation of task requirements. On the other hand, low confidence ratings could suggest insufficient comprehension of HARATIO features or ambiguities in the presentation and navigation between authoring activities.

The plot shows all participants completed at least 60% of the task and indicated moderate to high levels of confidence in their attempts. This result suggests an awareness and comprehension of the edit operations required to correct the errors and complete the scene; it also indicates participants were confident that the menu operations executed would result in the desired outcome.

Errors in completing tasks can be categorised as either ‘mistakes’ or ‘slips’ (Norman, 2013, p.170). A mistake is an error in the intended action caused by an incorrect goal or plan. A slip is an error in carrying out the intended action. Mistakes and slips can be further subclassified. According to Norman (pp.171–172), mistakes fall into ‘rule-based’, ‘knowledge-based’, or ‘memory-lapse’ categories. Rule-based mistakes occur when the user has diagnosed the situation correctly but then follows an incorrect course of action. Knowledge-based mistakes occur because the user has insufficient knowledge to formulate

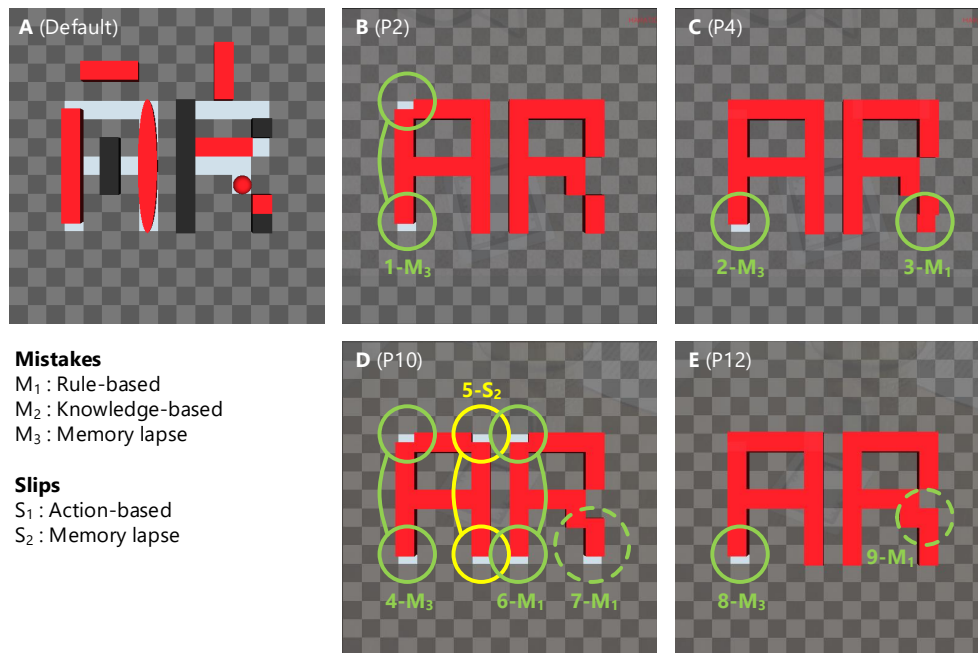


Figure 6.12: Task one errors (b-e). The default scene state is provided for reference in (a).

the correct plan, and memory-lapse mistakes occur because of a lapse in memory while assessing the situation, formulating a plan, or evaluating the result. Slips, on the other hand, can be categorised as either ‘action-based’ or ‘memory-lapse’ (p.171). Action-based slips relate to the incorrect action being performed while memory-lapse slips are similar to memory-lapse mistakes and refer to actions or evaluations not being carried out due to a lapse in memory.

As previously mentioned, four of the participants—one from the EO group and three from the IO group—achieved partial task success and had minor errors in their scenes (Figure 6.12). Analysis of these participants’ scenes and screen recordings suggest a combination of mistakes (shown in green) and slips (shown in yellow) contributed to this. Mistakes ranged from cubes being left in their default state (1- M_3 , 2- M_3 , 4- M_3 , and 8- M_3), cubes that required only a colour change additionally being scaled or removed (6- M_1 and 7- M_1), plane objects from the underlying silhouette being treated as cubes (3- M_1), and additional, superfluous cubes being placed outside zone boundaries (9- M_1). In the case of the slip (5- S_2), the participant correctly replaced the skewed cylinder with a cube but forgot to scale it to cover the entire length of the zone area.

While M_3 mistakes appear in each of the erroneous scenes, analysis of screen recordings offers a simple explanation. In each case, participants overlooked the cube and never interacted with it. The difference in zone coverage between Figure 6.12b/d and Figure 6.12c/e can be explained by the cube in zone two—the top of the A as illustrated in Figure 6.10—bleeding into zone one and filling the gap.

The M_1 mistakes in Figure 6.12d/e, shown with a dotted circle, highlight potential deficiencies in the implementation of HARATIO’s scale and translate operations that may have contributed to their occurrence. In both cases, the mistakes occur in zone ten, which, looking at the default task state in Figure 6.12a, could be completed by simply changing the colour of the bottom cube from black to red. However, the participant in each case

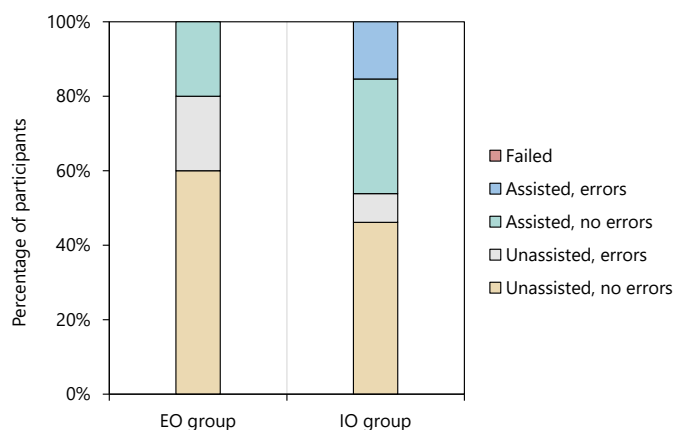


Figure 6.13: Breakdown of task one completion showing assistance, errors, and task failure

elected to delete the black cube and attempt to scale the red cube to fill the resulting gap. Per discussion in section 4.4.4.2, each increase in scale is equivalent to a factor of one grid cell. As scaling occurs uniformly, this results in the target object increasing or decreasing by half a grid cell in each x or y direction. As objects are aligned to grid cells based on their centre point, a cube scaled to an equivalent size of *two* will never be able to precisely occupy the two grid cells defined in zone ten; it will instead appear to occupy one grid cell and then half a cell at either end, leaving a visible gap at the top or bottom of the zone. Attempting to translate the object down half a cell to compensate would not help for the same alignment reason. This can be seen in the $7-M_1$ mistake in Figure 6.12d. The $9-M_1$ mistake in Figure 6.12e is similar except, in this instance, the participant has scaled the cube to an equivalent size of *three* so it occupies the space of the original black cube but also a redundant cell above. Although both are technically rule-based mistakes—the situation was appropriately diagnosed, but an incorrect course of action followed—the plan devised by these participants was reasonable and it is the implementation of the scale and translate operations that likely prevented it from being executed successfully. It could also be argued these errors were equally knowledge-based as the participants in question did not possess adequate understanding of the operations to work around this behaviour.

Cases where assistance was provided was also recorded during task attempts. An assist was deemed to be a situation in which a participant sought clarification or help from the test moderator regarding how to proceed with the task. This was either in terms of HARATIO assistance (i. e. how to use a particular operation) or task assistance. Regardless of the number of times a participant asked for assistance, only one case was recorded. A single EO participant and six IO participants were recorded as seeking assistance. This can be seen in Figure 6.13, which shows cases where assistance was provided as purple and green regions based on whether the resulting attempt contained errors. As assistance rates were lower among participants with prior HARATIO experience, H1b is rejected. As the graph illustrates, there were no recorded task failures or cases where participants prematurely gave up.

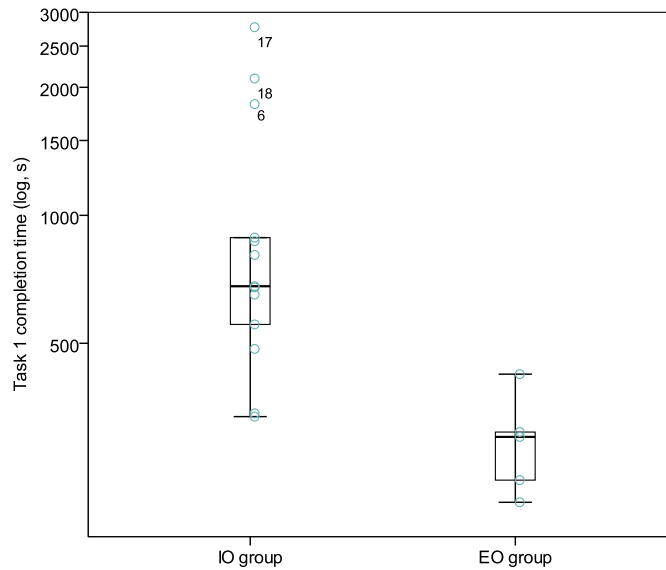


Figure 6.14: Task completion time (in seconds) by experience group. Data is shown in \log_{10} scale for clearer visualisation.

6.6.1.2 Efficiency

Task completion times were calculated from participants' first interaction with HARATIO following the task scene being loaded to selection of the 'finished' option within the slide over menu. As with the analysis of task times presented in Chapter 5, the geometric mean (GM) was used. Overall, task one was completed in a GM of 606 s ($SD = 709$ s). In terms of experience level, IO participants required a GM of 806 s ($SD = 750$ s) while EO participants required a GM of 288 s ($SD = 82$ s). Individual completion times for both experience groupings are presented in Figure 6.14. Examination of the graph reveals three outliers (6, 17, and 18) taking between three and four-and-a-half times longer than the overall GM; these participants were all over 61 years of age, retired, and were noted as being unfamiliar with the concept of working in a 3D environment.

A measure of task efficiency (E_t) can be derived from the ratio between completion rate and time on task as per the Common Industry Format (CIF) for Usability Test Reports. Using the formula expressed in equation 6.2, an indication of task success rate per unit of time can be determined.

$$E_t = \frac{CR}{GM_{min}} \quad (6.2)$$

The formula refers to the calculated completion rate (CR) presented in equation 6.1 and the GM task time in minutes (GM_{min}). The overall efficiency rate was 9.4% per minute. By experience level, for each minute spent on the task, participants in the EO group completed 20% while participants in the IO group completed 7%.

Though the results indicate a low overall efficiency in using HARATIO's radial menu to complete the task, the percentage values can only provide limited insight without comparisons to alternate menu designs. The results do, however, support H3 and suggest previous exposure to be beneficial in reducing cognitive effort as EO users were more than twice as efficient at completing the task. Familiarity with the interface would result in less time being spent searching the menu for appropriate operations or determining

how they function. This was supported by feedback from participants in the IO group who made comments during the task to the effect of being initially unsure how to begin editing objects; whether more than one object could be edited concurrently; and whether rotate, scale, and translate operations should be performed directly or indirectly. This last point was the subject of one participant's (P14) post-task comments with respect to confusion and repetitiveness that resulted from presenting these operations in separate menus:

“Clicking back after every move seemed very repetitive. Overall, not extremely hard to use, just a bit of a confusing interface.”

As discussed in section 4.2.2, a core design goal of the radial menu was to provide a focussed view of menu operations relevant to the immediate authoring task being performed. If the Edit activity was active, the menu would display items related to object editing; if the rotate operation was active, the menu would display items related to rotating the object, and so forth. Given most users would likely be accustomed to direct interaction with touch screen content, due to experience with modern mobile devices and applications, the need to frequently move between menus to access these operations independently would have been considered confusing and possibly *inefficient*. However, the participant's indication that the menu was ‘not extremely hard to use’ does suggest the simplification afforded by filtering menu items was effective and left them confident in the operations being performed—their rating of performance satisfaction in using it to complete the task was 6 out of 7.

6.6.1.3 Performance Satisfaction

An SEQ response was provided following the completion of the task and was recorded on a seven-point Likert scale where one represented ‘very difficult’ and seven ‘very easy’. The overall mean SEQ rating for the task was 4.5 ($SD = 1.47$). By HARATIO experience, participants in the IO group gave the task a mean rating of 4.15 ($SD = 1.52$) while participants in the EO group gave the task a mean rating of 5.4 ($SD = .89$). A stacked bar chart of SEQ scores, grouped by HARATIO experience, is provided in Figure 6.15. The chart reveals an even distribution of scores among IO participants compared with a neutral to positive skew for EO participants. The IO participants who considered the task more difficult by rating it below 4 made the following post-task remarks that provide some insight into the problems encountered:

- *“I was initially unsure what ‘translate’ meant.”*
- *“I expected to be able to translate by dragging an object directly, regardless of the menu [state]. The same with scale; I expected to be able to use a pinch-to-zoom gesture.”*
- *“I had initial difficulty separating movements of objects with movements of the background.”*
- *“[I] thought it was more intuitive to be able to scale and move through gestures and touch rather than [the] menu.”*

The difficulties identified relate to HARATIO's menu interface in terms of non-technical language use and user expectations with the functionality of Edit activity operations. The first comment highlights the need for interface designers to consider the use of

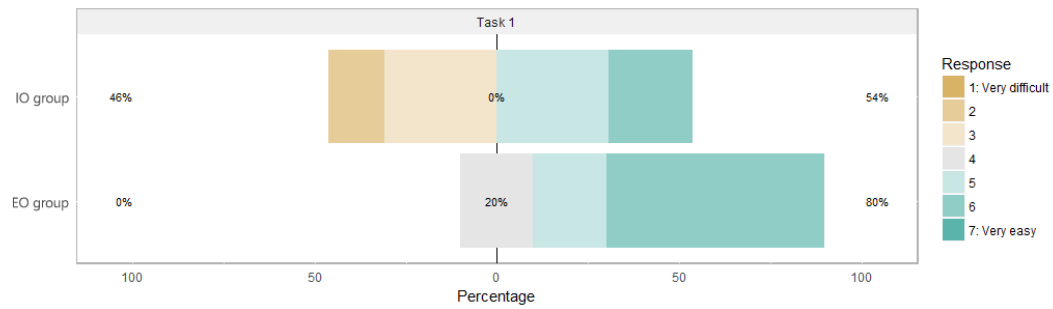


Figure 6.15: Distribution of task one SEQ responses by experience grouping.

terminology when labelling menu items. The term *translate* is unlikely to be familiar to those without 3D modelling experience, or possibly have an alternate meaning, and so may come across as ambiguous. The participant who made the comment sought assistance from the test moderator during the task with regards to its meaning. Upon explanation, the participant suggested perhaps ‘move’ would be a better term that was less susceptible to misinterpretation.

Justification for implementing indirect rotate and scale operations was covered in section 4.4.4.2, but it is inferred that established interaction experiences gained from the use of modern touch-based applications is likely to have contributed to participant confusion. This was also noted in the previous discussion in section 6.6.1.2 regarding the repetitiveness associated with consecutively moving between rotate, scale, and translate menus. Analysis of screen recordings shows multiple occasions where the *right* edit action was selected but the associated operation was initially attempted directly even though the menu provided an interaction affordance via the manipulation handle—resulting in an action-based slip. A follow-on observation discovered frequent attempts to mix or combine edit operations. For example, while the rotate operation was activated, attempts would be made to drag the object as if performing a translation. This speaks to the second remark above where the participant expressed an expectation that they should be able to perform *all* edit operations regardless of menu state.

Interestingly, one participant remarked that it ‘would be useful to be able to duplicate objects’. This feature already existed within the Edit activity as the Clone operation. As its menu item is located alongside other core edit operations, discoverability is no better or worse (see Figure 4.6 in Chapter 4). Therefore, such a comment indicates either the terminology *clone* was unclear or the feature was merely overlooked.

6.6.2 Task 2: Behaviour Script Editor

Results from the second task were used to evaluate the behaviour script editor in terms of enabling users without programming experience to effectively and efficiently add virtual object interactivity via the creation of behaviour scripts. As with the radial menu evaluation, analysis compared the effects of HARATIO experience and included measures of task completion, task errors, editor assistance, editor efficiency, and participant satisfaction.

6.6.2.1 Effectiveness

As with task one, task completion accounted for partial success so as not to dismiss attempts that contained minor errors. Given the variability with which behaviour scripts could be configured to complete the task goal, tasks were graded on a scale of 0 to 1: a score of 0 denoted a failed attempt, a score of 0.5 an attempt that was partially correct, and a score of 1 a successful attempt without errors. As per the task description in section 6.2.2, partial correctness was defined as an attempt in which at least one of the following was implemented:

- The tank object moved along the grey path
- The tank object changed colour to green when passing under the archway
- The tank object stopped on the red destination plate and changed colour back to red

Failure to satisfy at least one of these conditions was deemed a failed attempt. This approach proved simpler and more consistent than assigning points for individual parts of scripts given the variability in possible implementations. Aggregate completion rate (*CR*) was therefore determined using a modified version of equation 6.1 that multiplied the result by 100 instead of 10 based on completion being out of 1. As before, *n* is the number of participants and *r* the result obtained by participant *i* for the task:

$$CR = \left(\frac{\sum_{i=1}^n r_i}{n} \right) \times 100 \quad (6.3)$$

Overall completion rate for the second task was 89%. By HARATIO experience grouping, EO participants achieved a perfect completion rate of 100% while IO participants achieved an 85% (*SD* = .24) completion rate.

All participants achieved a score of 0.5 or better and created functional behaviours that were syntactically correct and executed successfully. As this includes all IO participants without programming experience, H4 is confirmed. While most stuck to a stepwise two-behaviour approach, comparable to the example provided in Table 6.1, seven IO participants opted to consolidate both behaviours into one script. This involved incorporating the archway colour change as part of the movement routine around the path. Participants with previous HARATIO experience (EO) instead utilised repeat actions for the tank movement parts that were recurrent. Various implementations of behaviour scripts suggest participants could grasp the conceptual differences between Triggers and Actions and arrange them accordingly to produce valid results, which further endorses H4. This is also supported by the high confidence ratings provided by participants following the task. The overall mean confidence rating was 6.5 (*SD* = .62). By experience level, both groupings were similar with EO participants recording a mean confidence rating of 6.8 (*SD* = .48) and IO participants recording a mean confidence rating of 6.38 (*SD* = .65).

The graph in Figure 6.16 plots confidence ratings against their corresponding task attempt score. Given there were only three possible completion scores, the display of quadrants is less useful, but the plot nevertheless demonstrates that successful task attempts—those that were awarded a score of 1—were followed by strong associated confidence levels. This implies such participants were comfortable with using the behaviour editor interface and confident their actions would lead to correct output.

The four participants from the IO group whose scripts contained errors likewise had

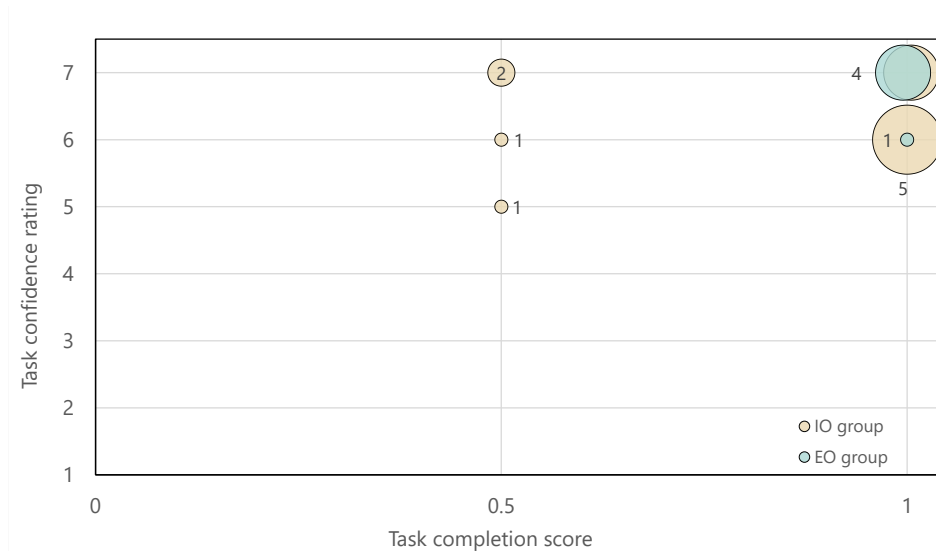


Figure 6.16: Relationship between task completion rate and perceived confidence level for task two. Numbers central or adjacent to bubbles indicate size (i. e. frequency).

high confidence levels that their attempts were correct. Analysis of these scripts, along with screen recordings, suggests the errors were comprised of slips and mistakes.

The slip (S_2) can be seen in Table 6.2a, which shows a behaviour script that is complete and correct other than the reversed ordering of the last two actions (lines 8 and 9). In this case, the participant had established the right plan but seemingly overlooked the ordering of these actions resulting in the tank changing colour (back to red) too soon—that is, before it reached the red destination plate.

The three mistakes, shown in Table 6.2b–d, include cases where the task goal conditions have been misinterpreted (M_2) or the participant has elected to complete the task in one behaviour (a perfectly valid option) while forgetting about the existence (and effect) of another (M_3). In Table 6.2c, the final Move Forward action (line 8) has been set to six cells instead of four, causing the tank to finish back where it started instead of on the red destination plate. In Table 6.2d, the participant has inserted unnecessary triggers to the start of the behaviour (lines 1 and 2) requiring a one second delay and tank interaction before any actions are performed. The first action (line 3) also unnecessarily changes the tank’s colour to orange, a result of possible confusion with the default colour action from the initial state of the first behaviour. Table 6.2b shows two scripts that ultimately conflict with one another. In this case, the participant had corrected the issue with the first behaviour—changing the colour from orange to green—but then proceeded to incorporate all functionality into the second behaviour, effectively rendering the first superfluous. The addition of a Wait trigger to the second behaviour (line 1 of B2) also caused the tank to change colour to green too soon as the first behaviour, running in parallel, operated with an effective delay of one second instead of four: line 1 of B1 (4 seconds) minus line 1 of B2 (3 seconds).

HARATIO’s behaviour test feature is a possible contributor to all described errors. The feature was utilised between 7–16 times by the erroneous scripts’ authors. As described in section 4.4.5.6, the Test Actions button, available on the Behaviour activity menu, sequentially executes all actions of the currently loaded behaviour script. It does not

Table 6.2: Erroneous behaviour scripts (Bn) from participant (P) task two attempts.

	Behaviour Script	COMMENT
(a) B2 P1	<ol style="list-style-type: none"> 1. When [the behaviour starts] 2. do [move forward 6 grid cells] 3. then [turn right 90°] 2. do [move forward 6 grid cells] 3. then [turn right 90°] 4. then [move forward 6 grid cells] 5. then [turn right 90°] 6. then [move forward 6 grid cells] 7. then [turn right 90°] 8. then [set colour to red] 9. then [move forward 4 grid cells] 10. end 	<p>S₂ Lines 8 and 9 reversed</p>
(b) B1 P2	<ol style="list-style-type: none"> 1. When [4 seconds have elapsed] 2. do [set colour to green] 3. end 	
B2	<ol style="list-style-type: none"> 1. When [3 seconds have elapsed] 2. do [move forward 6 grid cells] 3. then [turn right 90°] 4. then [move forward 3 grid cells] 5. then [set colour to green] 6. then [move forward 3 grid cells] 7. then [turn right 90°] 8. then [move forward 6 grid cells] 9. then [turn right 90°] 10. then [move forward 4 grid cells] 11. then [set colour to red] 12. end 	<p>M₃ Line 1 changed from default Always trigger Conflicts with B1</p>
(c) B2 P4	<ol style="list-style-type: none"> 1. When [the behaviour starts] 2. do [move forward 6 grid cells] 3. then [turn right 90°] 4. then [move forward 6 grid cells] 5. then [turn right 90°] 6. then [move forward 6 grid cells] 7. then [turn right 90°] 8. then [move forward 6 grid cells] 9. then [set colour to red] 10. end 	<p>M₂ Should be 4 grid cells</p>
(d) B1 P14	<ol style="list-style-type: none"> 1. When [1 second has elapsed] 2. and [the user taps on the object] 3. do [set colour to orange] 4. then [move forward 6 grid cells] 5. then [turn right 90°] 6. then [move forward 3 grid cells] 7. then [set colour to green] 8. then [move forward 3 grid cells] 9. then [turn right 90°] 10. then [move forward 6 grid cells] 11. then [turn right 90°] 12. then [move forward 4 grid cells] 13. then [set colour to red] 14. end 	<p>M₂ Triggers and initial action (Line 3) are unnecessary</p>

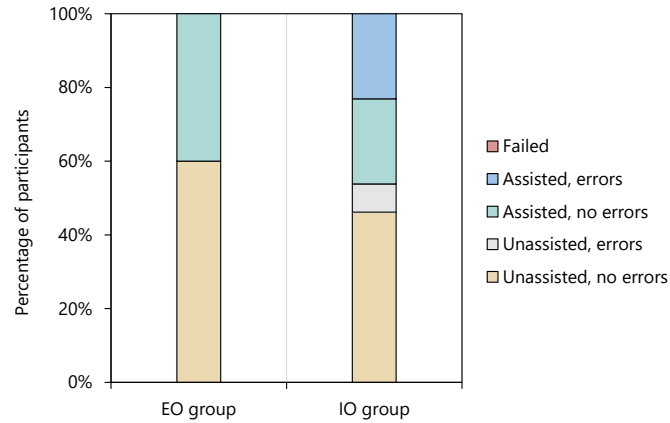


Figure 6.17: Breakdown of task two completion showing assistance, errors, and task failure

execute any triggers nor does it consider actions of other scripts. Tested in isolation, the second script (B2) in Table 6.2a would not appear erroneous. As the tank started out coloured red and the testing feature did not consider the actions of the first script, its colour would not visibly change and therefore the ordering slip would go unnoticed. The same applies for the second script (B2) in Table 6.2b. The conflict with the first script would not be apparent, and testing it would show successful path navigation with appropriate colour changes. The M_2 mistake in Table 6.2d may have also been partially caused by this operation as the unnecessary triggers would have been concealed during testing.

It is suspected the testing feature implementation may have prompted some participants to utilise a single script for the task rather than persist with the two as initially supplied; this is supported by post-task remarks made by one participant (P5):

“Confused by behaviour test and tried to do it all in 1 behaviour and not 2.”

By incorporating the bridge colour change into the movement routine of the tank around the path (shown in lines 4–6 of B2 in Table 6.2b), task progress could be evaluated without having to return to AR mode to view multiple scripts running concurrently. In total, seven attempts, including the two represented in Table 6.2, opted to use a single script for all task requirements. The implementation of the testing feature should be clarified or rethought in future iterations. A display layout depicting script relationships coupled with a test feature that evaluates all scripts together, as they would function when viewed live, would be one possible approach.

As with task one, cases where assistance was provided to participants was recorded during task attempts. Two EO participants and six IO participants sought assistance during the task. As before, this was in terms of clarifying the operation of the behaviour script editor or task requirements. A graph of task assistance rates, along with task completion, is provided in Figure 6.17. Also, as per the task one results, there were no recorded failures or cases where participants prematurely gave up.

6.6.2.2 Efficiency

Task completion times were calculated as per task one. Overall GM completion time was 529 s ($SD = 409$ s). EO participants completed the task in a GM of 312 s ($SD = 119$ s) while

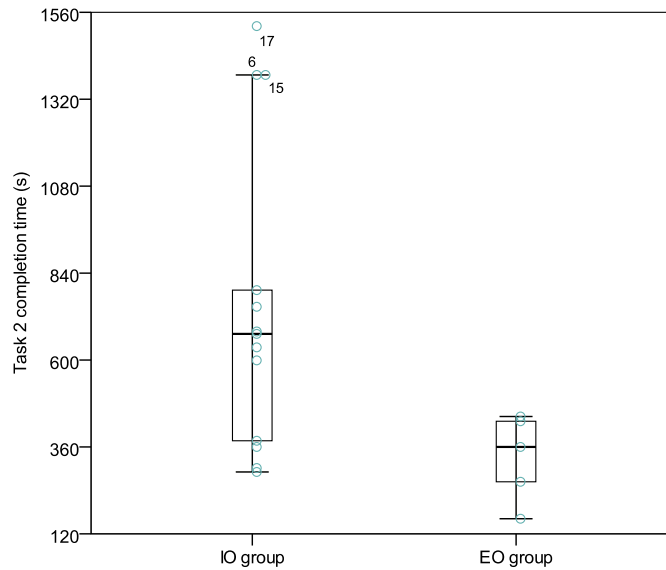


Figure 6.18: Task two completion time (in seconds) by experience group

IO participants required a GM of 649 s ($SD = 424$ s). Figure 6.18 shows completion times for participants in both groups. Like the previous task, examination of the graph reveals three outliers that required almost three times longer than the overall GM to complete the task. Two of these (6 and 17) were among the same group of outliers from task one and were over the age of 61 while the other was a student under the age of 40.

A measure of task efficiency was calculated using the same formula as in equation 6.2 with the adjusted CR formula from equation 6.3. Overall task efficiency was 10.1% per minute. In terms of HARATIO experience, for every minute spent on the task, IO participants completed 7.8% while EO participants completed 19.2%. Compared with task one, efficiency increased by 0.8% for the IO group and decreased by a reciprocal 0.8% for the EO group.

Like the previous task, use of the behaviour script editor achieved low overall efficiency. Further investigation into the differences between groupings suggested some IO participants found the term *behaviour* ambiguous as they were observed seeming initially unsure how to begin the task when faced with the default scene state and root menu. EO participants, familiar with the interface and behaviour operation, didn't face such issues. Furthermore, IO participants who sought assistance often queried the role and interaction between separate behaviour scripts. This was evidenced in post-task feedback as well as the strategy of consolidating all task functionality into a single behaviour—see the comment towards the end of section 6.6.2.1. While the raw percentages are difficult to derive too much meaning from without comparisons to alternate behaviour editor interfaces, the increased efficiency rate achieved by participants with HARATIO experience confirms H3 and indicates efficiency with the editor improves with experience. Further evaluations with participants from the EO group would be useful in substantiating this.

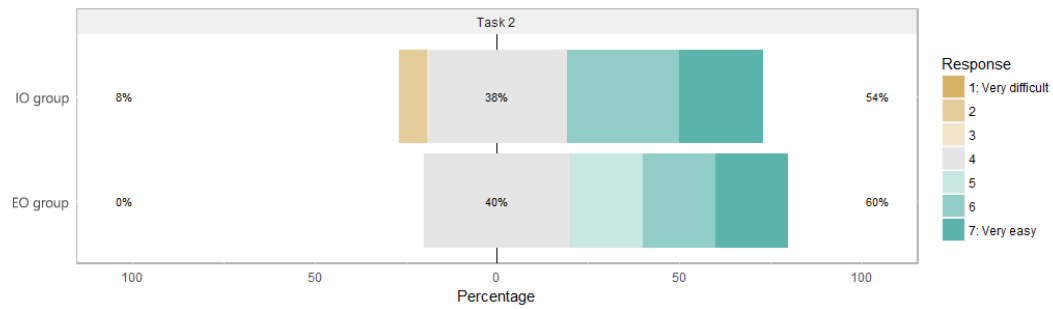


Figure 6.19: Distribution of task two SEQ responses by experience grouping

6.6.2.3 Performance Satisfaction

The overall mean SEQ rating for task two was 5.17 ($SD = 1.47$). By experience level, IO participants gave the task a mean rating of 5.15 ($SD = 1.57$) while EO participants gave it a mean rating of 5.2 ($SD = 1.3$). Based on the scores, H5 is confirmed as the IO group rating was greater than 4. The scores are comparable to the average SEQ score (4.8–5.1) reported by Sauro (2012) from data comprising more than 5000 responses.

A stacked bar chart of scores, grouped by HARATIO experience, is presented in Figure 6.19. Most participants, regardless of grouping, considered the task neither very difficult nor very easy and opted for a neutral response. The one participant (P17) who did consider the task difficult, with a rating of two, was also the same participant who had the longest completion time (Figure 6.18) and lowest efficiency, so this was unsurprising. Although the numbers in each group were different, the distribution of scores between them was similar compared with the first task, indicating less variation among participants with regards to the perceived difficulty.

Although efficiency with the use of the behaviour script editor was low, comments received post-task were generally favourable towards the interface and its operation:

- “The behaviour editor was very intuitive and well laid out.”
- “Very easy to use!”
- “Once I got past working out how to get [the tank] to move etc. it was straightforward and fun!”
- “I like the words-only view as a way to clarify the meaning of symbols.”
- “I really liked the ‘words’ to ‘code’ feature; it was good.”

The third comment relates to the observation discussed in section 6.6.2.2 regarding some IO participants appearing initially unsure how to begin defining behaviour scripts when presented with the default scene and root menu. As indicated by the comment, once it became clear that object interactivity was associated with the Behaviour activity, the process was perceived as being “straightforward and fun”. While experience with the interface would overcome such examples of uncertainty, the term *behaviour* could also be reassessed in a similar vein to the term *translate* discussed in section 6.6.1.3.

Notably, the last two comments singled out the behaviour translation feature. This feature was accessed at least once by ten (77%) IO participants and one (20%) EO participant. Out of the ten IO participants, the average use was twice per attempt. For those new to the concept of scripting, the ability to quickly confirm or clarify what an arrangement of Trigger and Action symbols would accomplish was perceived as a beneficial capability.

Such comments are encouraging and match the intended use of the feature. Conversations with these participants following the task also revealed a potential improvement to the presentation of the translation panel that would place it alongside the behaviour script rather than modally on top.

Additionally, a few other remarks focused specifically on comparisons with the previous radial menu task, suggesting these participants' perception of the behaviour script editor was based on a mental comparison with the menu interface:

- “Task 2 was much more intuitive.”
- “I found this one much easier than task 1.”

6.6.3 Observations Regarding the Effects of Age

Observations during participant sessions revealed an interesting trend among users of different age generations. In the event they were unsure how to progress or complete certain aspects of a task, younger users tended to be more willing to explore the interface before seeking assistance, trying different operations or combinations thereof and adopting a trial-and-error approach. Conversely, older users appeared more hesitant, almost fearful of making a mistake, and would seemingly stare at the screen contemplating what to do next or seek assistance over random interaction. In effect, they had more difficulty bridging the ‘gulf of execution’ (Norman, 2013, pp.38–40). This was especially true of the three 61-and-over participants.

The plots in Figure 6.20 show the relationship between task time and age resembles this behaviour: in general, older users took longer than younger users. A Kendall’s tau-b correlation was performed for both tasks to determine the extent of the relationship for participants across all age groups. The results revealed a medium positive correlation that was statistically significant for both task one ($T_b = .46, p = .014$) and task two ($T_b = .478, p = .011$).

The trial-and-error approach observed in younger users fits an analogy described by Frand (2000), who suggests those of a ‘Nintendo generation’ learn by constant trial-and-error rather than careful thought and planning—a *do* rather than *think* attitude ascribed with succeeding at playing video games and discovering secrets or new strategies. When presented with a unfamiliar device or piece of software, these users will immediately begin using it, pushing buttons and trying available options rather than studying documentation or considering what approach might be best.

Prensky’s (2001) digital native versus digital immigrant idea expands on this. Prensky argues those that have grown up with technology—the so-called Nintendo generation—process information in a fundamentally different way to those that have not. Consequently, this leads to different approaches to learning (to use a tool such as HARATIO). Prensky suggests digital immigrant users would be more inclined to seek out the manual rather than assume the program itself will provide the instruction. While the inclusion of the visual notification system in HARATIO (section 6.1.5) was intended to provide enhanced feedback and basic explanation of possible next interactions or clarification of selection behaviour, it may have simply been overlooked. One participant (P2) did comment that the hints disappeared too early with no obvious way to review them—“*Hints disappeared too early. There doesn’t appear to be any way to get them back.*”—so the delivery and presentation of

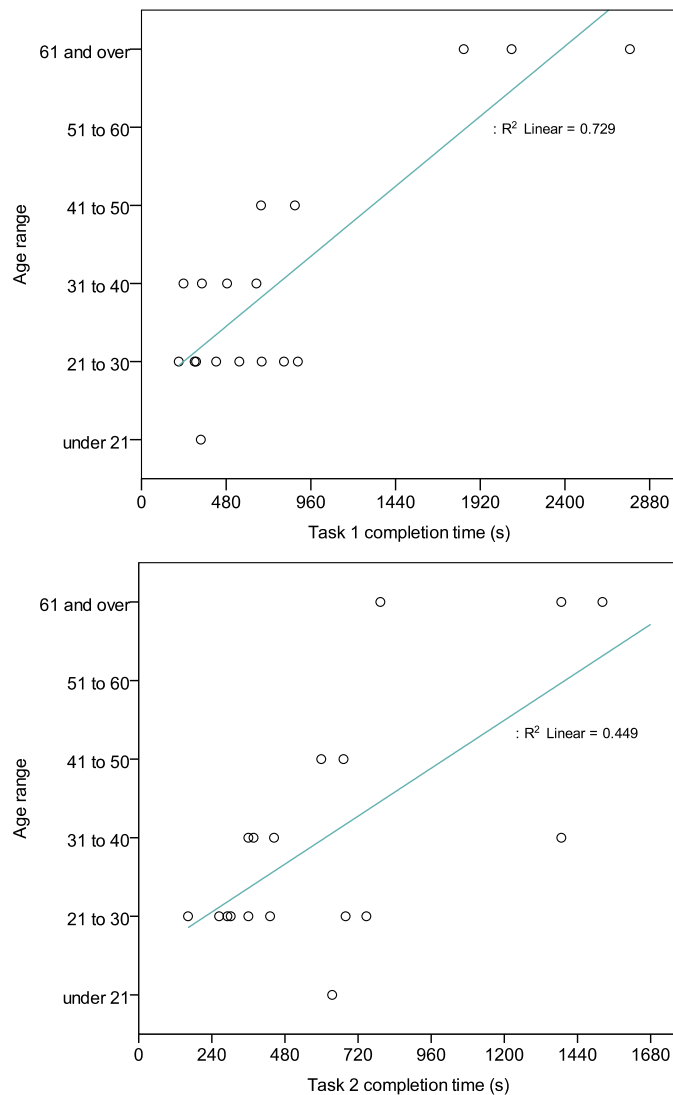


Figure 6.20: Plot of age group against task completion time for task one (top) and two (bottom).

this element warrants further evaluation. Prensky's notion might also explain why those in the older age groups appeared more hesitant to explore the interface. Although a brief overview was provided at the beginning of the session, it was intentionally lacking in detail and was delivered at a reasonably fast pace, leaving little time for details to sink in. For these users, a just-in-time or exploratory learning style may not have been ideal.

Although there is debate as to the validity of the digital native versus digital immigrant argument—see Bennett et al. (2008) and Selwyn (2009) as examples—the view that there is a difference between the way users of different age groups approach learning provides an interesting, if not plausible, explanation for some of the behavioural trends observed and is something to be mindful of when considering the user experience.

Even though a tool like HARATIO aims to provide an interface suitable for those without specific skillsets, each user will ultimately adopt a different method when learning to use it. For some, this might simply be a block of time spent exploring (discovery-based learning) while for others it may mean some form of written or instructor-led guidance; others still may prefer an entirely different approach. Ideally, once the core concepts have been learnt,

they can be transferred to other aspects of the interface through similar affordances. The efficiency improvements demonstrated by EO participants over IO participants for both tasks provides an empirical example of this.

6.7 Summary

This chapter has evaluated HARATIO's radial menu and behaviour script editor interfaces with a view to understanding how well they support novice users and the kinds of authoring tasks typical of a mobile AR authoring tool. Users comprising inexperienced and experienced groups participated in a formal user study in which two tasks were attempted. The distinction between groupings was based on previous experience with both HARATIO and programming. The first task asked participants to correct errors in an existing AR scene by using the radial menu to execute appropriate operations within Create and Edit activities. The second task asked participants to complete a series of behaviour scripts for an existing AR scene using the Behaviour script editor. A set of observations and reflections were derived from the results.

The success rate for the first task was similar between groups; experienced participants performed marginally better and required less assistance using the menu compared with inexperienced participants. Both groups indicated above-average confidence ratings. Observations highlighted issues with the design and functionality of core Edit activity operations—rotate, scale, and translate—with many inexperienced participants expecting to be able to perform these directly and routinely confused as to why this was not possible. This was further compounded by the inability to execute multiple edit actions concurrently within the menu as each needed to be accessed and used separately. SEQ responses regarding performance satisfaction subsequently reflected this. These issues were not as pronounced with participants familiar with HARATIO, who were more than twice as efficient in completing the task requirements and recorded fewer errors in doing so.

The second task produced a wider gap in success rate but recorded lower overall completion times compared to task one; experienced participants remained more than twice as efficient. Confidence ratings were found to be similarly high between groups. Varying behaviour script implementations were observed beyond those anticipated from the task description, and all participants produced syntactically valid and functional scripts that executed successfully. This suggested a level of comprehension regarding the appropriate arrangement of Trigger and Action symbols to produce valid results. Experienced participants commonly incorporated basic looping while inexperienced participants consolidated multiple behaviours into a single script. Behaviour consolidation was suspected to be a consequence of ambiguity with the interaction between multiple behaviour scripts and the limited operation of HARATIO's script testing feature, which could lead to script errors going unnoticed.

Seventy-seven percent of participants from the inexperienced group utilised the script translation feature at least once compared to only one from the experienced group. Use of the term *behaviour* to denote access to the Behaviour activity and script editor was also queried based on observations of some participants seeming initially unsure how to begin. Nonetheless, performance satisfaction ratings for the editor were above average for both

experience groups.

Observations during both tasks noted differences in the way participants of various age generations approached the use of HARATIO. Younger participants favoured carefree trial-and-error methods while older participants were more hesitant and preferred considered approaches involving thought and planning.

Conclusion

This thesis has presented an investigation of mobile AR authoring tools suitable for inexperienced users. In accordance with the concept of AR 2.0 (Schmalstieg et al., 2011) discussed in Chapter 1, a future of wide-scale mobile AR adoption will likely involve regular users using their own handheld devices to both consume and contribute interactive AR content with minimal effort. If regular users are to become effective authors, they will require easy-to-use tools that support authoring without any presumption of technical expertise in AR, programming, or 3D modelling. This research has been motivated by the desire to better understand the kinds of user interfaces necessary to support such tools as well as the interaction techniques that will underpin the user experience.

As discussed in section 2.7, commercial authoring solutions already exist in the form of SDKs and software to facilitate AR creation, but they require a degree of technical knowledge and access to a PC class machine to effectively use. They are therefore unsuitable for authoring in situ and are not optimised for touch interaction. While such tools will undoubtedly continue to evolve, the development of HARATIO within this thesis has enabled the exploration of appropriate authoring experiences that natively suit handheld devices and support a novice audience.

The remainder of this chapter summarises the contributions of this thesis based on the research questions, discusses limitations of the data derived from evaluations, and provides a list of possible future research directions emanating from this work.

7.1 Contributions

Guided by the overall research objective described in Chapter 1, this thesis has examined novice authoring solutions using off-the-shelf handheld devices. In doing so, the research has contributed to a better understanding of mobile AR authoring interfaces and experiences by addressing three questions: is there an ideal handheld form factor for mobile AR? can the complexities of AR authoring be simplified for mobile device users? and what interface factors should be considered to provide a suitable user experience for novices?

7.1.1 Ideal Form Factor for Mobile AR

Chapter 3 presented an initial user study into the current state of handheld devices with the goal of understanding how end user experience may be affected by physical size. The process began with a survey of available hardware suitable for mobile AR and was followed by an up-to-date classification of form factor categories comprising smartphone, mini tablet, and tablet. Devices typifying each category were then evaluated over the course of seven interactive AR tasks. Usability results concluded the smartphone and mini tablet form factors to be most usable. The tablet form factor was effected by size

and weight issues that overshadowed any advantage offered by a larger screen. Using a tablet was consequently more challenging due to interaction space instability caused by difficulties maintaining a secure grip with one hand while performing interactions with the other. However, a tablet can offer advantages in terms of performance satisfaction and interaction accuracy so long as weight and sub-optimal device orientation are not overwhelming factors. Overall, the mini tablet form factor offered the best end user AR experience and was the most subjectively preferred.

Chapter 5 found mobile AR authoring to be possible on devices from all three form factor categories. The addition of a freeze technique within HARATIO resolved interaction stability issues observed in Chapter 3 and enabled the tablet device to be used effectively without its size or weight dominating the experience. While small devices may not be suitable for large-scale or complex authoring tasks, for simple changes, quick prototypes, or reviewing existing scenes, they are certainly sufficient. As before, the mini tablet was subjectively preferred overall and recorded the highest performance satisfaction rating. Its usability score in the AR-focussed HARUS instrument demonstrated its balance between adjacent form factors in terms of content comprehensibility and manipulability. Based on results obtained during this research, the mini tablet represents an ideal form factor for mobile AR use.

7.1.2 Simplification of AR Authoring Complexities

The results from Chapter 3 were used to inform the design of a prototype handheld AR authoring tool called HARATIO. The tool, described in Chapter 4, explored user interface solutions for simplifying the complexities associated with transitioning into and out of authoring modes, exposing authoring options, and facilitating object scripting for non-programmers. Results gathered during this research have demonstrated that complex authoring tasks can be simplified for effective use on a range of mobile devices.

Transitioning Between AR and Author Modes

HARATIO provided a clear separation between live AR and author modes via an abstraction based on a play/pause metaphor reminiscent of interfaces for controlling media playback. Selecting the pause option froze the current AR view in place and enabled authoring tasks to be performed with virtual content fixed in its last registered position. Selecting the play option restored the view to a live AR experience. Both options were only displayed when applicable. From the user's perspective, the process was seamless and required no knowledge of technical AR concepts to use. The effectiveness of this metaphor was demonstrated in Chapters 5 and 6 where participants frequently used it to transition between AR and author modes while previewing and creating task content.

Exposing Authoring Options

Authoring operations were structured into a taxonomy of distinct authoring activities. An adaptation of a radial menu facilitated access to each activity and focussed user attention on distinct aspects of the creation process. The menu implemented a multi-menu hierarchy via a collapsible structure that dynamically reconfigured itself to efficiently utilise available

screen space. Menu configuration was further supported by a *visual breadcrumb* overlay that provided feedback on previous selections.

As shown in Chapter 5, the menu design was usable on handheld form factor categories ranging from a smartphone to a tablet. Chapter 6 evaluated the menu between users with and without AR and HARATIO experience. Results demonstrated similar effectiveness but a benefit to efficiency and performance satisfaction with experience. Participant confidence in using the menu was above average regardless of experience.

While the menu demonstrated the ability to effectively provide access to authoring options, observations identified conflicts between expectations with the way certain operations were performed as well as the structural organisation of operations into separate menus. These expectations were considered partly influenced by the behaviour of modern mobile applications. Though these concerns didn't prevent the menu being used to perform authoring tasks, they do highlight the need for menu designs to consider existing paradigms and the influences of established mobile application interactions.

Object Scripting for Non-programmers

The implementation of a visual drag-and-drop script editor supported the creation of per object interactivity (*behaviours*). The editor presumed no programming experience and employed a mix of assistive labels and draggable graphical symbols to form conditional expressions formatted as simple sentence phrases. Symbols comprised *triggers* and *actions*—one or more triggers specified conditions that, when satisfied, would execute one or more actions. To further aid users in understanding script semantics, translation into full natural language sentences was also integrated.

Feedback from Chapter 5 indicated the behaviour script editor was well-received and considered beneficial to non-programmers. Chapter 6 subsequently evaluated the editor with novices (non-programmers) as well as a small subset of users with prior HARATIO experience to evaluate effectiveness and learnability. All participants created functional behaviour scripts that executed successfully. Various implementations were produced beyond those anticipated indicating a level of comprehension regarding the appropriate arrangement of script components: novice participants frequently consolidated multiple scripts while experienced participants incorporated basic looping. Script translation was used twice on average by more than 75% of novices with feedback indicating it was beneficial in clarifying symbol meaning. Given the results, the complexities associated with defining object interactivity were demonstrated to be manageable by novice users within the constraints of a mobile platform.

Usability Benchmark

The usability results reported in Chapter 5 showed HARATIO to be usable across all three handheld form factors evaluated. In each case, scores obtained via the System Usability Scale (SUS) and Handheld AR Usability Scale (HARUS) were greater than a prescribed benchmark (70.14) determined from a sizeable pool of existing usability data.

With a minimum mean SUS score of 71.78 and a minimum mean HARUS score of 72.36, future handheld AR authoring tools may wish to consider comparisons against the usability scores of HARATIO. By establishing a benchmark score for novice-focussed

handheld AR authoring in this thesis, other tools with similar goals of simplifying the authoring experience can benefit.

7.1.3 Interface Considerations for a Novice User Experience

Based on the data collected throughout this research, a few factors have emerged that should be considered when designing AR authoring interfaces for handheld devices. These will help ensure the user experience provided is suitable. Many of these considerations align with established usability heuristics for user interfaces (Nielsen, 1994); however, there are cases where the existing list does not suitably cover the conditions relevant to a mobile AR authoring domain. The list below highlights cases where the considerations conform to existing heuristics (listed by their code from p.156 of Nielsen's work) and where there is scope for expansion.

View Freezing

The user evaluation discussed in Chapter 3 highlighted issues surrounding the use of larger tablet devices for mobile AR. Orienting a large handheld in the kinds of poses often required when tracking targets results in fatigue, underperformance, and interaction space instability. As implemented in HARATIO (Chapter 4) based on the research discussed in section 2.4, the inclusion of a freeze technique in which the AR tracker and device camera are temporarily paused allows the device to be reoriented to a less fatiguing pose with persistent interaction stability. Chapter 5 demonstrated the effectiveness of this technique in resolving issues with tablet use.

Consequently, mobile AR authoring solutions should adopt appropriate methods to provide interaction stability and remove, or at the very least lessen, manipulability fatigue. Participant ranking of hardware factors considered important for mobile AR further supports the benefits. In Chapter 3, where no freeze technique was available, 'ergonomics' and 'weight' were considered the two *most* important factors. Conversely, in Chapter 5, where a freeze technique was available, 'ergonomics' and 'weight' were considered the *least* two important factors. Although tablet devices benefit most from this, it is equally applicable to all form factors. Given the differences inherent in the user experience of mobile AR authoring applications, a relevant existing usability heuristic pertaining to view freezing is absent. The findings from the user evaluations performed in this thesis submit view freezing as an additional heuristic beyond the ten from Nielsen that should be considered when designing usable mobile AR interfaces.

Clear and Consistent Labelling

Established by discussion in Chapter 4, the use of menu item labels is beneficial to users unfamiliar with a system. Accordingly, label wording needs to be carefully considered so as not to unintentionally obfuscate the message or inadvertently affect discoverability of the underlying feature. Early iterations of HARATIO's radial menu design did not clearly indicate the number of behaviour scripts attached to objects or the method of creating new scripts. Participant feedback in Chapter 5 identified instances of confusion regarding this operation. The menu design was subsequently revised to clarify script association as

well as indicate how new scripts could be added: both occurred through clearer labelling.

Consistency between label messaging should also be maintained as users may not read every label before performing operations. The noted misunderstanding of the Repeat Action configuration discussed in Chapter 5 demonstrated an example where the dominant label relating to the value being configured required users read a supplementary label above to fully understand the effect of changes.

It is also easy to regress into the habit of using nomenclature unfamiliar to users outside of the related discipline when considering label text. Chapter 6 identified cases where certain label wording was perceived as unclear or ambiguous, which resulted in the utility of the associated feature being questioned. Rotate, scale, and translate (RST) are standard terms throughout 3D modelling tools, but they may be unfamiliar to those without modelling experience; the word *translate*, for example, was suggested to be replaced with a more descriptive alternative like *move*.

The principles of Nielson's 'language' (A2) and 'consistency' (A4) heuristics apply here in which systems should adopt terms and phrases familiar to users and be consistent in their use. The discussion above has highlighted cases throughout this thesis that demonstrates these are equally valid for mobile AR authoring interfaces.

Visual Feedback

While clear and consistent labels help users discover and access features, they do not assist with communicating outcomes or providing information in the event of unexpected issues.

Participant feedback discussed in Chapter 5 revealed a desire for more visual feedback. A notification system was implemented in Chapter 6 to address this by displaying timely messages comprising relevant hints or information on noteworthy events. However, as later participant feedback indicated, the presentation, duration, and reviewability of any finite messaging system need to be considered when pursuing this approach.

Of equal importance is clearly communicating the state and effect of operations. As described in Chapter 4, HARATIO implemented measures to assist with conveying object manipulation state by showing overlaid gimbal widgets; the radial menu related the effect of performing these operations within its adaptive core. Mitigating screen occlusion when dragging new objects into a scene was achieved via a callout that appeared above the finger contact point to reveal the obscured screen area beneath.

The user experience evaluations performed in this thesis support the use of established heuristics related to 'feedback' (A5) and 'good messages' (A8) when designing interfaces for mobile AR. However, further consideration should be given to feedback within a live AR view, such as communicating the current state of tracking. As described in section 4.2.2 and Figure 4.6 of Chapter 4, one approach explored within HARATIO's radial menu was the adaption of the menu core to indicate the state of fiducial marker tracking and only switch to appropriate interaction options when applicable. Integrating feedback into the menu in this fashion increases its visibility to users.

One Novice User Does Not Equal Another Novice User

As observed in Chapter 6, different age generations adopted different approaches when using HARATIO. Older participants exhibited more hesitancy and caution towards navi-

gating the interface and were more likely to seek assistance before performing authoring operations. Conversely, younger participants were willing to explore and learn by trial-and-error. The observations advise considering that there are different *levels* of novice users with different learning methodologies. While this thesis did not investigate age-related differences in detail, the anecdotal evidence highlights the need to adopt coherence and consistency within an interface to promote skill transfer as well as closely consider how the gulfs of execution and evaluation (Norman, 2013) apply to different generations.

While the previous considerations will help with this, it is also worth noting ‘help and documentation’ (A10) as the last of the ten usability heuristics. Though exploratory learning styles may preference trial-and-error over the consultation of manuals, the inclusion of on-demand assistance—either (digitally) integrated into the user experience or provided separately—should be available to provide help if necessary.

7.2 Limitations

The results discussed throughout this thesis were based on empirical findings from a series of user evaluations. This section acknowledges limitations with the design and structure of these evaluations that ultimately restricted the conclusions able to be derived.

Experimental Environment

All user evaluations were performed in a controlled indoor laboratory environment. While this was necessary for practical (the use of fiducial markers) and consistency reasons, it is not representative of the variety of locations in which mobile AR authoring is likely to occur. Consequently, evaluations were not able to consider potential effects on the user experience that may result from authoring outdoors, for example.

Participation Sample Bias

Apart from the inexperienced participant sample used for the evaluation in Chapter 6, all volunteers were drawn exclusively from undergraduate and postgraduate cohorts in the School of Computer Science, Engineering, and Mathematics at Flinders University. Many of these volunteers had passing knowledge of AR, this research, and other computer science concepts. This may have resulted in their performance and/or responses being subject to an implicit level of bias.

Moreover, most participants were aged between 21 and 30. Users in this age bracket are less likely to be regarded as typical novice users given their ‘millennial’ status and ‘digital native’ stereotype in which they are posited to have grown up surrounded by similar technologies (Prensky, 2001). In line with the observations discussed in Chapter 6 regarding differences between age generations, a more thorough investigation including additional age groups and volunteers outside the university would be warranted.

Handheld Devices

The physical devices used during user evaluations were selected based on their embodiment of different handheld form factor categories available at the time (see Chapter 3). However,

given the prevalence of mobile device ownership, user attitudes towards these devices may have been subconsciously influenced by preferences towards personal devices or particular platforms—the *PC (Windows) versus Macintosh* fanaticism that has seemingly migrated to *Android versus iOS*. This is not an easy problem to address save for the use of custom hardware or completely obfuscating the underlying system. Alternative methodologies, such as ‘Wizard of Oz’, offer other potential testing avenues to explore for interface usability studies that would help mitigate such biases.

7.3 Future Work

Several avenues for future work have been identified over the course of this research. These directions represent possible pathways for further exploration of HARATIO as well as alternative mobile AR authoring solutions.

Field Studies

As discussed in section 7.2, user studies reported on in this thesis were performed in a controlled laboratory environment with defined sets of tasks and goals. While attempts were made in Chapter 5 to include tasks that were more open-ended, promoting interface exploration and simulating realistic scenarios, additional studies performed in the field would offer further insights into general usability issues and interface design considerations not apparent in a laboratory. Such environments would also add unpredictability to authoring and require users deal with environmental conditions including obstacles, noise, lighting, and workspace distractions.

Field studies would ideally require the implementation of a tracking solution that does not rely on fiducial markers nor require environment pre-preparation.

Content Creation

The way in which virtual content is created within a tool like HARATIO is also an area for future exploration. For 3D objects, this may involve direct modelling support or scanning physical objects from the surrounding environment. In the case of modelling, it is unlikely novice users will possess the skills required, so solutions will have to consider how this can be achieved in a suitable manner. The investigation of Rod et al. (2016) into novice 3D modelling interfaces for multi-touch devices provides one example. While this thesis limited content to simple 3D models, content types beyond that could be considered. Media types including text, images, video, and audio would enable more varied and unique scenes to be created.

Alternative Scripting Mechanisms

Alternate approaches to defining interactivity could be explored as a supplement or replacement to the drag-and-drop approach implemented within HARATIO. Instead of defining object behaviours indirectly by way of meaningfully arranging various symbols, a more direct approach could be investigated whereby users create macro-like recordings of interactions and the system interprets them accordingly. Moving an object to a specific point in

a scene might involve dragging it to the desired location or physically moving the host device.

Emerging Form Factors

The emergence of new mobile form factors deserves regular evaluation to maintain up-to-date awareness of suitable mobile AR platforms. This thesis has shown on AR usability between large and small handhelds (Chapter 3), so having an appreciation for form factor characteristics and differences will benefit the user experience.

New ‘wearable’ form factors may prove appropriate as input/output auxiliaries for mobile AR authoring tools, such as HARATIO, running on a main device. The nascent smartwatch market has the potential to become an interesting accessory that may allow for certain interface elements to be assigned to a secondary screen or provide alternate means of controlling specific operations. This would require careful consideration in terms of appropriate interaction so as not to cause unnecessary confusion.

Such devices could also be used to explore other means of providing feedback in addition to visual elements. Haptic feedback provides a mechanism for alerting users to events without requiring they look at a screen, and is often variable in terms of ferocity and pattern. This would enable various combinations to be used for different events and could be used to explore interactions that feel more tangible or provide further cues as to what the tool is doing. Haptics would also be interesting to implement into scripting grammars to allow users to integrate tangible feedback into their own AR scenes.

Storage, Collaboration, and Distribution

Future mobile authoring solutions could explore mechanisms for seamlessly storing, sharing, and distributing AR content, both between multiple users and multiple devices owned by the same user. As described in Chapter 5, authoring could begin on a mobile device in situ and then continue later with a different device. Rather than users manually managing this process, solutions could be explored to support the transition between devices automatically. Collaborative authoring could also be investigated whereby many users, perhaps with different device form factors, are able to share and contribute to a scene just by being in the same physical space (collocated). A similar proximity approach would be interesting to explore for scene distribution in which available AR content is retrieved automatically as soon as a user enters the relevant location. These scenarios will all require the development of suitable interfaces that hide complexity and provide functional yet simple user experiences.

7.4 Final Remarks

As AR continues to evolve and become more mainstream, interest from regular users looking to create their own interactive AR experiences will require the availability of authoring solutions that de-emphasise complexity and assume no technical expertise. This thesis has demonstrated the viability of mobile AR authoring for novice users by investigating interfaces for off-the-shelf handheld devices. While the solutions explored

contribute to a better understanding of suitable user experiences, further research is required to fully realise the potential of AR in everyday scenarios.



Handheld Device Survey

The following lists data collected from a survey conducted of consumer handheld devices in 2014. Devices were included in the survey based on their suitability for base level mobile AR use. Minimum requirements were deemed a rear-facing camera and a mobile operating system supporting third party applications. This would enable, at a minimum, a fiducial marker-based AR application to be used. Only distinct screen sizes from each manufacturer were counted.

Survey data of distinct handheld devices suitable for AR use (ca. 2014)

MANUFACTURER	MODEL	SCREEN SIZE (IN)
Acer	Iconia A1	7.9
Acer	Iconia W3	8.1
Acer	Iconia W5	10.1
Acer	Iconia A3	10.1
Acer	Iconia W7	11.6
Apple	iPhone 4S	3.5
Apple	iPhone 5S	4
Apple	iPad Mini	7.9
Apple	iPad Air	9.7
ASUS	PadFone	4.3
ASUS	Nexus 7	7
ASUS	Fonepad	7
ASUS	MeMO Pad HD	8
ASUS	Transformer Pad	10.1
ASUS	VivoTab	11.6
Dell	Venue 8	8
Dell	Venue 11	10.8
HP	7 Plus	7
HP	8	7.85
HTC	Desire C	3.5
HTC	One V	3.7
HTC	Desire X	4
HTC	One Mini	4.3
HTC	Desire 601	4.5
HTC	One	4.7
HTC	One M8	5
HTC	Desire 816	5.5
HTC	One Max	5.9

Survey data of distinct handheld devices suitable for AR use (ca. 2014)

MANUFACTURER	MODEL	SCREEN SIZE (IN)
Huawei	Ascend Y300	4
Huawei	Ascend G6	4.5
Huawei	Ascend P6	4.7
Huawei	Ascend G730	5.5
Huawei	Ascend Mate	6.1
Huawei	MediaPad 10	10.1
Lenovo	Yoga Tablet 8	8
Lenovo	ThinkPad 8	8.3
Lenovo	Yoga Tablet 10	10.1
Lenovo	ThinkPad Helix	11.6
LG	Optimus L2II	3.2
LG	L40	3.5
LG	Optimus L5II	4
LG	Optimus L7II	4.3
LG	L70	4.5
LG	Nexus 4	4.7
LG	Nexus 5	4.95
LG	G2	5.2
LG	G3	5.5
LG	G Flex	6
LG	G PAD	8.3
Microsoft	Surface 2	10.6
Microsoft	Surface Pro 3	12
Motorola	E	4.3
Motorola	G	4.5
Motorola	X	4.7
Nokia	Lumia 520	4
Nokia	Lumia 1020	4.5
Nokia	Lumia 625	4.7
Nokia	Lumia 930	5
Nokia	Lumia 1320	6
Nokia	Lumia 2520	10.1
Samsung	Galaxy Music	3
Samsung	Galaxy Gio	3.2
Samsung	Galaxy Mini 2	3.27
Samsung	Galaxy Ace Plus	3.65
Samsung	Omnia W	3.7
Samsung	Galaxy Ace 3	4
Samsung	Galaxy S4 Mini	4.3
Samsung	Galaxy S2	4.5
Samsung	Galaxy S3	4.8
Samsung	Galaxy S4	5
Samsung	Galaxy S5	5.1
Samsung	Galaxy Note	5.29

Survey data of distinct handheld devices suitable for AR use (ca. 2014)

MANUFACTURER	MODEL	SCREEN SIZE (IN)
Samsung	Galaxy Note 2	5.55
Samsung	Galaxy Note 3	5.7
Samsung	Galaxy Tab 3	7
Samsung	Galaxy Tab 3	8
Samsung	Galaxy Tab S	8.4
Samsung	Nexus 10	10.055
Samsung	Galaxy Tab 3	10.1
Samsung	Galaxy Tab S	10.5
Samsung	Galaxy NotePRO	12.2
Sony	Xperia E Single	3.5
Sony	Xperia M	4
Sony	Xperia Z1 Compact	4.3
Sony	Xperia SP	4.6
Sony	Xperia M2	4.8
Sony	Xperia Z1	5
Sony	Xperia Z2	5.2
Sony	Xperia Z Ultra	6.4
Sony	Xperia Tablet Z2	10.1

B

Description of Triggers and Actions

In reference to discussion of the Behaviour language in Chapter 4, this section provides expanded descriptions of triggers (section 4.4.5.2) and actions (section 4.4.5.3) added to the language for the research presented in this thesis. Each trigger/action was selected on the basis of being useful or necessary for task completion throughout user evaluations of HARATIO covered in Chapters 5 and 6. Where cited, some actions made use of the iTween animation framework (Berkebile, 2011) to support their operation.

B.1 Triggers

Always

This trigger was the simplest trigger available. It contained no conditions and was always satisfied. It was intended for use when a behaviour's actions were to be performed immediately as the soon as the processing routine evaluated the parent behaviour script.

Wait

This trigger served to delay action execution by n seconds. The delay countdown began when the behaviour was first evaluated. When the countdown reached zero, the trigger condition was satisfied. The range of configurable durations was between 1 and 30 seconds and could be adjusted in 1 second increments. The `Wait` trigger could be invalidated by other triggers, which would cause the delay countdown to restart.

Distance

This trigger used data from the AR tracker to evaluate the distance between the device and fiducial marker. Distance evaluation was performed by analysing the magnitude between vectors representing the marker and device camera. For the trigger condition to be satisfied, the distance (in cm) had to be less than or equal to the distance configured by the user. The range of configurable values was 20–100 cm and could be adjusted in 5 cm increments. The `Distance` trigger could invalidate other triggers if the calculated distance moved beyond the configured value.

Tap

This trigger relied on interaction with the device and could be used to determine when the parent object was selected. The trigger condition was satisfied as soon as a single tap on the parent object was detected. The trigger could also be invalidated by other triggers, which would cause its state to be reset and allow for another interaction to be captured. This

was useful in situations where multiple triggers had to be satisfied before a behaviour's actions were performed; for example, a case where a device needs to be within a certain distance of the marker before selection can occur.

Focus

This trigger was used to determine whether the host device was currently targeting (focusing on) a virtual object. As discussed in section 4.3, when HARATIO was in AR mode, a crosshair was displayed in the centre of the screen. The Focus trigger cast a ray that originated from the crosshair to determine whether the device was focussing on the object. Only when the ray cast intersected the object would the trigger become satisfied. As with the Distance trigger, the Focus trigger could invalidate other triggers in the event object focus was lost.

B.2 Actions

Move Forward

This action moves the parent object forward in the direction it is currently facing by a specified number of grid cells as configured by the user. For the evaluations discussed in this thesis, the range of configurable values was set to 1–15 and adjustments could be made in 1 cell increments. The speed at which the object moves can be controlled by the Set Speed action. Movement is performed with the aid of the iTween animation framework.

Turn Left

This action rotates the parent object ninety degrees to the *left* of its up axis. The object remains stationary while rotating. Object rotation is performed with the aid of the iTween animation framework.

Turn Right

This action functions as per the Turn Left action but rotates the parent object to the *right* of its up axis instead of the left.

Set Speed

This action altered the speed at which movement-based actions (Move Forward, Turn Left, and Turn Right) will operate. Changes are immediate and affect all behaviour scripts attached to the parent object. For this thesis, speed values could be set between 1 (the default) and 10 and adjusted in increments of 1. The values were used in conjunction with the iTween animation framework to modify movement speed.

Colour

This action changed the colour of the parent object. All colour swatches available in the colour menu of the Edit activity (section 4.4.4.1) could be selected when configuring the action.

Repeat

This action provided basic looping functionality by repeating execution of all actions that were positioned *before* it in the sequence. The repetition count was limited to between 1 and 50 iterations and could be adjusted in increments of 1. It was therefore impossible for a user to create an infinite loop. The functionality of the Repeat action can be considered analogous to a `for` statement used within typical programming languages.

Stop All

This action was a terminal action that immediately halted all running behaviour scripts on the parent object. The action was useful in situations where multiple behaviour scripts were attached to an object and the occurrence of a certain condition or interaction would need to result in all interactivity ceasing.



Ethics Approval for Experiment 1

Any experiment involving human participants from Flinders University requires appropriate ethics approval from the university's Social and Behavioural Research Ethics Committee (SBREC). A suitable ethics application (project number 6409) pertaining to the experiment discussed in Chapter 3 was submitted and approved prior to the commencement of any recruitment or testing. A copy of the final approval notice appears below:

Lawrence Sambrooks

From: Human Research Ethics
Sent: Wednesday, 12 March 2014 11:56 AM
To: Lawrence Sambrooks; Brett Wilkinson
Subject: 6409 SBREC Ethics - Final Ethics Approval (12 March 2014)

Dear Lawrence,

The Chair of the [Social and Behavioural Research Ethics Committee \(SBREC\)](#) at Flinders University considered your response to conditional approval out of session and your project has now been granted final ethics approval. Your ethics final approval notice can be found below.

FINAL APPROVAL NOTICE

Project No.:

Project Title:

Principal Researcher:

Email:

Approval Date: Ethics Approval Expiry Date:

The above proposed project has been **approved** on the basis of the information contained in the application, its attachments and the information subsequently provided.

RESPONSIBILITIES OF RESEARCHERS AND SUPERVISORS

1. Participant Documentation
Please note that it is the responsibility of researchers and supervisors, in the case of student projects, to ensure that:

- all participant documents are checked for spelling, grammatical, numbering and formatting errors. The Committee does not accept any responsibility for the above mentioned errors.
- the Flinders University logo is included on all participant documentation (e.g., letters of Introduction, information Sheets, consent forms, debriefing information and questionnaires – with the exception of purchased research tools) and the current Flinders University letterhead is included in the header of all letters of introduction. The Flinders University international logo/letterhead should be used and documentation should contain international dialling codes for all telephone and fax numbers listed for all research to be conducted overseas.
- the SBREC contact details, listed below, are included in the footer of all letters of introduction and information sheets.

1



Ethics Approval for Experiment 2

Any experiment involving human participants from Flinders University requires appropriate ethics approval from the university's Social and Behavioural Research Ethics Committee (SBREC). A suitable ethics application (project number 6881) pertaining to the experiment discussed in Chapter 5 was submitted and approved prior to the commencement of any recruitment or testing. A copy of the final approval notice appears below:

Lawrence Sambrooks

From: Human Research Ethics
Sent: Wednesday, 17 June 2015 2:39 PM
To: Lawrence Sambrooks; Brett Wilkinson
Subject: 6881 SBREC final approval notice (17 June 2015)

Importance: High

Dear Lawrence,

The Chair of the [Social and Behavioural Research Ethics Committee \(SBREC\)](#) at Flinders University considered your response to conditional approval out of session and your project has now been granted final ethics approval. This means that you now have approval to commence your research. Your ethics final approval notice can be found below.

FINAL APPROVAL NOTICE

Project No.:

Project Title:

Principal Researcher:

Email:

Approval Date: Ethics Approval Expiry Date:

The above proposed project has been **approved** on the basis of the information contained in the application, its attachments and the information subsequently provided.

RESPONSIBILITIES OF RESEARCHERS AND SUPERVISORS

1. Participant Documentation
Please note that it is the responsibility of researchers and supervisors, in the case of student projects, to ensure that:

- all participant documents are checked for spelling, grammatical, numbering and formatting errors. The Committee does not accept any responsibility for the above mentioned errors.
- the Flinders University logo is included on all participant documentation (e.g., letters of Introduction, information Sheets, consent forms, debriefing information and questionnaires – with the exception of purchased research tools) and the current Flinders University letterhead is included in the header of all letters of introduction. The Flinders University international logo/letterhead should be used and documentation should contain international dialling codes for all telephone and fax numbers listed for all research to be conducted overseas.

1



Ethics Approval for Experiment 3

A modification to the ethics application (project number 6881) pertaining to the experiment discussed in Chapter 5 was submitted and approved prior to recruiting any participants for the experiment discussed in Chapter 6. A copy of the final approval appears below:

Lawrence Sambrooks

From: Human Research Ethics
Sent: Friday, 8 January 2016 10:14 AM
To: Lawrence Sambrooks; Brett Wilkinson
Subject: 6881 SBREC modification No.1 approval notice (8 January 2016)

Importance: High

Follow Up Flag: Follow up
Flag Status: Completed

Dear Lawrence,

The Chairperson of the [Social and Behavioural Research Ethics Committee \(SBREC\)](#) at Flinders University has reviewed and approved the modification request that was submitted for project 6881. A modification ethics approval notice can be found below.

MODIFICATION (No.1) APPROVAL NOTICE

Project No.:

Project Title:

Principal Researcher:

Email:

Modification Approval Date: Ethics Approval Expiry Date:

I am pleased to inform you that the modification request submitted for project 6881 on the 9 December 2015 has been reviewed and approved by the SBREC Chairperson. Please see below for a list of the approved modifications. Any additional information that may be required from you will be listed in the second table shown below called 'Additional Information Required'.

Approved Modifications	
Extension of ethics approval expiry date	
Project title change	
Personnel change	
Research objectives change	
Research method change	X
Participants – addition +/- change	

1



Task 1 Reference Sheet for Experiment 2

Below is the task reference sheet provided to participants before attempting the first task discussed in Chapter 5. In addition to the reference sheet, participants were also provided with a fiducial marker and handheld device running HARATIO.

Task 1
Ensure you've watched the video of the task before attempting to recreate it. A description of the task is provided below for reference.

The diagram shows a 17x17 grid with columns and rows numbered 1 to 17. An orange square is located at the intersection of column 5 and row 5. A dashed square path is drawn around the orange square, with arrows indicating a clockwise direction. The path starts at the top of the orange square (row 5, column 5), moves up to row 13, then right to column 13, then down to row 5, and finally left to column 4. The orange square itself is located between columns 4 and 6 and rows 4 and 6.

Description
An orange square with a volume of 4 is initially placed at grid location 5,5. It is configured to move along a square path (depicted above) in a clockwise direction at a speed of 2. It starts moving immediately and repeats this path a total of 5 times.
The user must select the square by tapping on it before it finishes moving. Once the square has been successfully selected, it changes colour to green and stops moving.



Task 2 Reference Sheet for Experiment 2

Below is the task reference sheet provided to participants before attempting the second task (scenario exercise) discussed in Chapter 5. In addition to the reference sheet, participants were also provided with a fiducial marker and handheld device running HARATIO.

Scenario 2

You are part of a group that is creating an augmented reality game for an upcoming university project. The group wants to visualise the design of a particular level and you have been tasked with creating a quick AR mock-up. The level involves a tank crossing a bridge and moving under an archway to navigate to a safe zone.

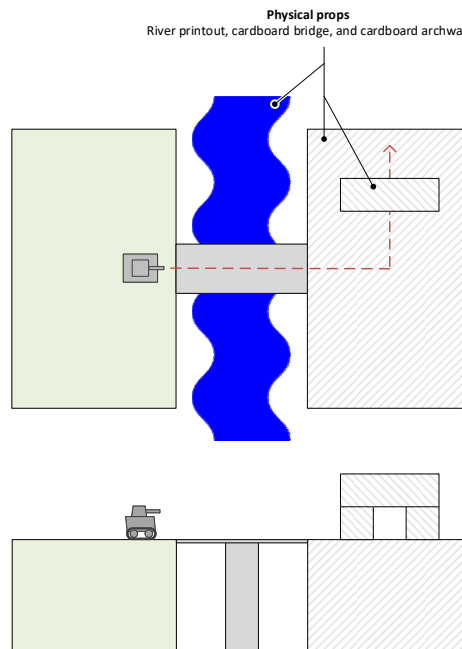
The level will need to be designed around three real world objects: a printed graphic representing a river, a 3D cardboard box representing one of the cliffs, and a 3D cardboard model representing the archway. These will have been placed in the working environment for you.

The cliffs should be located either side of the river and a bridge should connect them. The bridge will additionally need a support pylon to hold it up. The **tank** should be placed at the entrance to the bridge, on the virtual cliff side, and be configured to cross it and pass under the archway. The tank should begin moving automatically after 5 seconds.

As the final game assets are still being created, you can use **cubes** for the cliffs and archway, a **plane** for the bridge, and a **cylinder** for the support pylon.

A few sketches of the level have been provided to you from one of the other team members that should help to clarify the design and location of objects.

*Hint: consider assigning the **occlusion** colour to virtual objects that represent real world objects.*





Task Materials for Experiment 3

Below are the two task reference sheets and post-task questionnaire form provided to participants as part of the experiment discussed in Chapter 6.

TASK 1

Before attempting this task, you will need to open the associated file named **Task 1.arscene**

You will be presented with a scene that should spell out the abbreviation 'AR'. However, something has gone wrong and the scene is in a bit of a mess! Some of the blocks are misplaced, some are incorrectly coloured, and others don't seem to belong at all.

Your task is to correct the issues by making use of the various application modes accessible through the menu. Existing blocks can be edited via the *edit* mode and new blocks can be created via the *create* mode.

Use the grey silhouette of the letters as a guide for correctly placing the blocks.

The final scene should look like the image below:



TASK 2

Before attempting this task, you will need to open the associated file named Task 2.arscene

You will be presented with a scene that shows a red tank at the beginning of a grey path. The tank should follow the path around to the red square at the end.

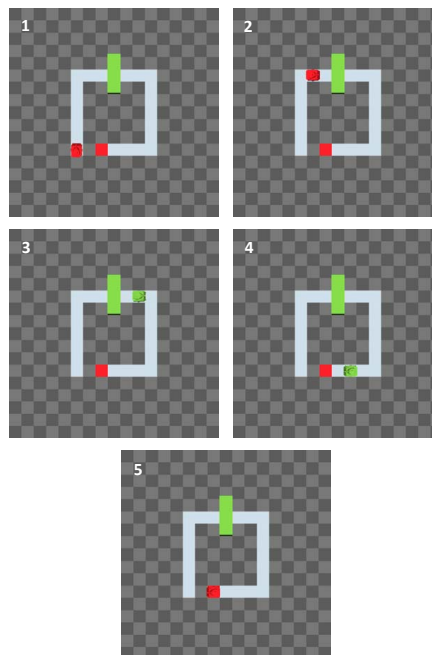
The tank has already been configured with two behaviours but the first is misconfigured and the second is incomplete—you will need to fix them.

The first behaviour is complete but is misconfigured. This behaviour will change the colour of the tank when it passes under the green bridge. It should change the colour to green to match the bridge, but it is currently configured for a different colour. Edit this behaviour so it functions correctly.

The second behaviour has no actions. Add appropriate actions that enable the tank to follow the path. Remember to change the colour of the tank back to red once it reaches the red square at the end of the path.

Resume the application to AR mode to check you have fixed both the behaviours.

The final flow of the scene should be as below (left-to-right, top-to-bottom):



Post-task questionnaire

Project number: 6881

Task

ID

For each question, please circle the number that best reflects your answer.

1. Overall, I found this task...

Very difficult

Very easy

1 2 3 4 5 6 7

2. How confident are you that you completed the task successfully?

Not at all
confident

Extremely
confident

1 2 3 4 5 6 7

3. Any other feedback/comments?

References

- Abawi, D. F., Dörner, R., Haller, M. and Zauner, J. (2004), Efficient mixed reality application development, in '1st European Conference on Visual Media Production (CVMP)', pp. 289–294.
- Apple Inc. (2017), 'ARKit'. Accessed: July 2017.
URL: <https://developer.apple.com/arkit>
- Arshad, H., Chowdhury, S. A., Chun, L. M., Parhizkar, B. and Obeidy, W. K. (2016), 'A freeze-object interaction technique for handheld augmented reality systems', *Multimedia Tools and Applications* **75**(10), 5819–5839.
- Arth, C. and Schmalstieg, D. (2011), 'Challenges of large-scale augmented reality on smartphones', *Graz University of Technology, Graz* pp. 1–4.
- AS/NZS (2006), Software engineering—Software product Quality Requirements and Evaluation (SQuaRE)—Common Industry Format (CIF) for usability test reports, ISO/IEC 25062:2006 (R2016), Standards Australia/Standards New Zealand.
- Auer, S. and Dick, E. (2007), When does a difference make a difference? A snapshot on global icon comprehensibility, in 'International Conference on Human-Computer Interaction', Springer, pp. 3–12.
- Azuma, R. T. (1993), 'Tracking requirements for augmented reality', *Communications of the ACM* **36**(7), 50–51.
- Azuma, R. T. (1997), 'A survey of augmented reality', *Presence: Teleoperators and virtual environments* **6**(4), 355–385.
- Azuma, R. T., Bailiot, Y., Behringer, R., Feiner, S., Julier, S. and MacIntyre, B. (2001), 'Recent advances in augmented reality', *IEEE Computer Graphics and Applications* **21**(6), 34–47.
- Bai, H., Lee, G. A. and Billinghamurst, M. (2012), Freeze view touch and finger gesture based interaction methods for handheld augmented reality interfaces, in 'Proceedings of the 27th Conference on Image and Vision Computing New Zealand', ACM Press, pp. 126–131.
- Bailly, G., Lecolinet, E. and Nigay, L. (2007), Wave menus: Improving the novice mode of hierarchical marking menus, in 'IFIP Conference on Human-Computer Interaction', Springer, pp. 475–488.
- Bangor, A., Kortum, P. and Miller, J. (2009), 'Determining what individual SUS scores mean: Adding an adjective rating scale', *Journal of usability studies* **4**(3), 114–123.
- Bangor, A., Kortum, P. T. and Miller, J. T. (2008), 'An empirical evaluation of the System Usability Scale', *International Journal of Human-Computer Interaction* **24**(6), 574–594.

- Bell, B., Feiner, S. and Höllerer, T. (2001), View management for virtual and augmented reality, in 'Proceedings of the 14th annual ACM symposium on User interface software and technology - UIST '01', ACM, pp. 101–110.
- Bennett, S., Maton, K. and Kervin, L. (2008), 'The 'digital natives' debate: A critical review of the evidence', *British journal of educational technology* **39**(5), 775–786.
- Berger, M.-O. (1997), Resolving occlusion in augmented reality: a contour based approach without 3D reconstruction, in 'Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on', IEEE, pp. 91–96.
- Berkebile, B. (2011), 'iTween'. Accessed: November 2017.
URL: <http://itween.pixelplacement.com>
- Berning, M., Kleinert, D., Riedel, T. and Beigl, M. (2014), A study of depth perception in hand-held augmented reality using autostereoscopic displays, in 'Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on', IEEE, pp. 93–98.
- Bier, E. A., Stone, M. C., Pier, K., Buxton, W. and DeRose, T. D. (1993), Toolglass and magic lenses: the see-through interface, in 'Proceedings of the 20th annual conference on Computer graphics and interactive techniques', ACM, pp. 73–80.
- Billinghamurst, M., Bowskill, J., Dyer, N. and Morphett, J. (1998), 'Spatial information displays on a wearable computer', *IEEE Computer Graphics and Applications* **18**(6), 24–31.
- Billinghamurst, M. and Henrysson, A. (2006), 'Research directions in handheld AR', *IJVR* **5**(2), 51–58.
- Billinghamurst, M., Kato, H. and Poupyrev, I. (2001), 'The MagicBook: a transitional AR interface', *Computers & Graphics* **25**(5), 745–753.
- Biocca, F. A. and Rolland, J. P. (1998), 'Virtual eyes can rearrange your body: Adaptation to visual displacement in see-through, head-mounted displays', *Presence: Teleoperators and Virtual Environments* **7**(3), 262–277.
- Bonnet, D. and Appert, C. (2011), Sam: the swiss army menu, in '23rd French Speaking Conference on Human-Computer Interaction', ACM Press, p. 5.
- Bowman, D. A. and Hodges, L. F. (1997), An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments, in 'Proceedings of the 1997 symposium on Interactive 3D graphics', ACM, pp. 35–ff.
- Breen, D. E., Whitaker, R. T., Rose, E. and Tuceryan, M. (1996), 'Interactive occlusion and automatic object placement for augmented reality', *Computer Graphics Forum* **15**(3), 11–22.
- Brook, M. (2014), 'Fireplug 3D Model'. Accessed: November 2017.
URL: <https://archive3d.net/?a=download&id=f5e9c6f1>
- Brooke, J. et al. (1996), 'SUS – a quick and dirty usability scale', *Usability evaluation in industry* **189**(194), 4–7.

- Cakmakci, O., Ha, Y. and Rolland, J. P. (2004), A compact optical see-through head-worn display with occlusion support, *in* 'Third IEEE and ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 16–25.
- Callahan, J., Hopkins, D., Weiser, M. and Shneiderman, B. (1988), An empirical comparison of pie vs. linear menus, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 95–100.
- Cartoon tank* (2014). Accessed: November 2017.
 URL: <https://free3d.com/3d-model/tank-68936.html>
- Caudell, T. P. and Mizell, D. W. (1992), Augmented reality: an application of heads-up display technology to manual manufacturing processes, *in* 'System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on', Vol. ii, pp. 659–669 vol.2.
- Cockburn, A., Ahlström, D. and Gutwin, C. (2012), 'Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse', *International Journal of Human-Computer Studies* **70**(3), 218–233.
- Cohé, A., Dècle, F. and Hachet, M. (2011), tBox: a 3D transformation widget designed for touch-screens, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', ACM Press, pp. 3005–3008.
- Cooper, S., Dann, W. and Pausch, R. (2003), Teaching objects-first in introductory computer science, *in* 'ACM SIGCSE Bulletin', Vol. 35, ACM, pp. 191–195.
- Cubillo, J., Martin, S., Castro, M. and Boticki, I. (2015), 'Preparing augmented reality learning content should be easy: UNED ARLE - an authoring tool for augmented reality learning environments', *Computer Applications in Engineering Education* **23**(5), 778–789.
- Delail, B. A., Weruaga, L. and Zemerly, M. J. (2012), CAViAR: Context aware visual indoor augmented reality for a university campus, *in* '2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology', IEEE Computer Society, pp. 286–290.
- Drascic, D. and Milgram, P. (1996), Perceptual issues in augmented reality, *in* 'Proceedings-Spie The International Society For Optical Engineering', SPIE INTERNATIONAL SOCIETY FOR OPTICAL, pp. 123–134.
- Edwards, E. K., Rolland, J. P. and Keller, K. P. (1993), Video see-through design for merging of real and virtual environments, *in* 'Virtual Reality Annual International Symposium, 1993., 1993 IEEE', pp. 223–233.
- Eitsuka, M. and Hirakawa, M. (2013), Authoring animations of virtual objects in augmented reality-based 3D space, *in* 'Advanced Applied Informatics (IIAIAI), 2013 IIAI International Conference on', IEEE, pp. 256–261.
- Feiner, S., MacIntyre, B., Haupt, M. and Solomon, E. (1993), Windows on the world: 2D windows for 3D augmented reality, *in* 'Proceedings of the 6th annual ACM symposium on User interface software and technology - UIST '93', ACM, pp. 145–155.

- Feiner, S., MacIntyre, B., Hollerer, T. and Webster, A. (1997), A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment, in 'Wearable Computers, 1997. Digest of Papers., First International Symposium on', IEEE, pp. 74–81.
- Feiner, S. and Shamash, A. (1991), Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers, in 'Proceedings of the 4th annual ACM symposium on User interface software and technology', ACM, pp. 9–17.
- Fiala, M. (2005a), ARTag, a fiducial marker system using digital techniques, in 'Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on', Vol. 2, pp. 590–596 vol. 2.
- Fiala, M. (2005b), Comparing ARTag and ARToolkit Plus fiducial marker systems, in 'Haptic Audio Visual Environments and their Applications, 2005. IEEE International Workshop on', IEEE, p. 6.
- Fiala, M. (2010), 'Designing highly reliable fiducial markers', *IEEE Transactions on Pattern analysis and machine intelligence* **32**(7), 1317–1324.
- Finstad, K. (2006), 'The system usability scale and non-native english speakers', *Journal of usability studies* **1**(4), 185–188.
- Finstad, K. (2010), 'Response interpolation and scale sensitivity: Evidence against 5-point scales', *Journal of Usability Studies* **5**(3), 104–110.
- Fitts, P. M. (1954), 'The information capacity of the human motor system in controlling the amplitude of movement.', *Journal of experimental psychology* **47**(6), 381.
- Francone, J., Bailly, G., Lecolinet, E., Mandran, N. and Nigay, L. (2010), Wavelet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode, in 'Proceedings of the International Conference on Advanced Visual Interfaces', ACM, pp. 173–180.
- Frاند, J. L. (2000), 'The information-age mindset changes in students and implications for higher education', *Educause review* **35**, 14–25.
- Fuhrmann, A., Hesina, G., Faure, F. and Gervautz, M. (1999), 'Occlusion in collaborative augmented environments', *Computers & Graphics* **23**(6), 809–819.
- Gandy, M. and MacIntyre, B. (2014), Designer's augmented reality toolkit, ten years later: implications for new media authoring tools, in 'Proceedings of the 27th annual ACM symposium on User interface software and technology', ACM, pp. 627–636.
- Gausemeier, J., Freund, J., Matysczok, C., Bruederlin, B. and Beier, D. (2003), Development of a real time image based object recognition method for mobile AR-devices, in 'Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa', ACM, pp. 133–139.
- Gervautz, M. and Schmalstieg, D. (2012), 'Anywhere interfaces using handheld augmented reality', *Computer* **45**(7), 26–31. P.
- Gimeno, J., Morillo, P., Orduña, J. M. and Fernández, M. (2013), 'A new AR authoring tool using depth maps for industrial procedures', *Computers in Industry* **64**(9), 1263–1271.

- Grasset, R., Langlotz, T., Kalkofen, D., Tatzgern, M. and Schmalstieg, D. (2012), Image-driven view management for augmented reality browsers, in '2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)', pp. 177–186.
- Grasset, R., Looser, J. and Billinghurst, M. (2005), A step towards a multimodal ar interface: A new handheld device for 3D interaction, in 'Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 206–207.
- Guyen, S., Feiner, S. and Oda, O. (2006), Mobile augmented reality interaction techniques for authoring situated media on-site, in 'Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 235–236.
- Hallaway, D., Feiner, S. and Höllerer, T. (2004), 'Bridging the gaps: Hybrid tracking for adaptive mobile augmented reality', *Applied Artificial Intelligence* **18**(6), 477–500. P
- Haller, M., Stauder, E. and Zauner, J. (2005), AMIRE-ES: Authoring mixed reality once, run it anywhere, in 'Proceedings of the 11th International Conference on Human-Computer Interaction (HCI)' , Vol. 2005.
- Hampshire, A., Seichter, H., Grasset, R. and Billinghurst, M. (2006), Augmented reality authoring: generic context from programmer to designer, in 'Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments', ACM, pp. 409–412.
- Haringer, M. and Regenbrecht, H. T. (2002), A pragmatic approach to augmented reality authoring, in 'Mixed and Augmented Reality, 2002. ISMAR 2002. Proceedings. International Symposium on', IEEE, pp. 237–245.
- Henrysson, A. and Billinghurst, M. (2007), Using a mobile phone for 6 DOF mesh editing, in 'Proceedings of the 8th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI', ACM, pp. 9–16.
- Henrysson, A., Billinghurst, M. and Ollila, M. (2005a), Face to face collaborative AR on mobile phones, in 'Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on', IEEE, pp. 80–89.
- Henrysson, A., Billinghurst, M. and Ollila, M. (2005b), Virtual object manipulation using a mobile phone, in 'Proceedings of the 2005 international conference on Augmented tele-existence - ICAT '05', ACM, pp. 164–171.
- Henrysson, A., Ollila, M. and Billinghurst, M. (2005), Mobile phone based AR scene assembly, in 'Proceedings of the 4th international conference on Mobile and ubiquitous multimedia', ACM, pp. 95–102.
- Hoffman, D. M., Girshick, A. R., Akeley, K. and Banks, M. S. (2008), 'Vergence-accommodation conflicts hinder visual performance and cause visual fatigue', *Journal of Vision* **8**(3).
- Höllerer, T. and Feiner, S. (2004), 'Mobile augmented reality', *Telegeoinformatics: Location-Based Computing and Services*. Taylor and Francis Books Ltd., London, UK **21**.

- Höllerer, T., Feiner, S., Hallaway, D., Bell, B., Lanzagorta, M., Brown, D., Julier, S., Baillot, Y. and Rosenblum, L. (2001), 'User interface management techniques for collaborative mobile augmented reality', *Computers & Graphics* **25**(5), 799–810.
- Höllerer, T., Feiner, S., Terauchi, T., Rashid, G. and Hallaway, D. (1999), 'Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system', *Computers & Graphics* **23**(6), 779–785. P
- Hong, J. (2013), 'Considering privacy issues in the context of Google glass', *Commun. ACM* **56**(11), 10–11.
- Hopscotch Technologies (2017), 'Hopscotch – Learn to Code Through Creative Play'. Accessed: July 2017.
URL: <https://www.gethopscotch.com>
- Howland, K. and Good, J. (2015), 'Learning to communicate computationally with Flip: A bi-modal programming language for game creation', *Computers & Education* **80**(0), 224–240.
- Hürst, W. and Van Wezel, C. (2013), 'Gesture-based interaction via finger tracking for mobile augmented reality', *Multimedia Tools and Applications* **62**(1), 233–258.
- Icons8 LLC (2017), 'Icons8'. Accessed: June 2017.
URL: <https://icons8.com>
- Jain, P., Manweiler, J. and Choudhury, R. R. (2015), OverLay: Practical mobile augmented reality, in 'Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '15', ACM, pp. 331–344.
- Julier, S., Lanzagorta, M., Baillot, Y., Rosenblum, L., Feiner, S., Höllerer, T. and Sestito, S. (2000), Information filtering for mobile augmented reality, in 'Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on', pp. 3–11. P
- Karaman, A., Erisik, D., Incel, O. D. and Alptekin, G. I. (2016), 'Resource usage analysis of a sensor-based mobile augmented reality application', *Procedia Computer Science* **83**, 300–304.
- Kato, H. and Billinghurst, M. (1999), Marker tracking and HMD calibration for a video-based augmented reality conferencing system, in 'Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on', pp. 85–94.
- Kerber, F., Lessel, P., Mauderer, M., Daiber, F., Oulasvirta, A. and Krüger, A. (2013), Is autostereoscopy useful for handheld AR?, in 'Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia', MUM '13, ACM, New York, NY, USA, pp. 4:1–4:4.
- Kim, H., Reitmayr, G. and Woo, W. (2013), 'IMAF: in situ indoor modeling and annotation framework on mobile phones', *Personal and ubiquitous computing* **17**(3), 571–582.
- Kiyokawa, K., Kurata, Y. and Ohno, H. (2000), An optical see-through display for mutual occlusion of real and virtual environments, in 'Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on', pp. 60–67.

- Klein, G. and Drummond, T. (2004), Sensor fusion and occlusion refinement for tablet-based AR, in 'Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on', IEEE, pp. 38–47.
- Klein, G. and Murray, D. (2007), Parallel tracking and mapping for small AR workspaces, in '2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality', pp. 225–234.
- Klein, G. and Murray, D. (2009), Parallel tracking and mapping on a camera phone, in '2009 8th IEEE International Symposium on Mixed and Augmented Reality', pp. 83–86.
- Knoedel, S. and Hachet, M. (2011), Multi-touch RST in 2D and 3D spaces: Studying the impact of directness on user performance, in '3D User Interfaces (3DUI), 2011 IEEE Symposium on', pp. 75–78.
- Koutsourelakis, C. and Chorianopoulos, K. (2010), 'Icons in mobile phones comprehensibility differences between older and younger users', *Information Design Journal* **18**(1), 22–35.
- Kurkovsky, S., Koshy, R., Novak, V. and Szul, P. (2012), Current issues in handheld augmented reality, in 'Communications and Information Technology (ICCIT), 2012 International Conference on', IEEE, pp. 68–72.
- Kurtenbach, G. and Buxton, W. (1994), User learning and performance with marking menus, in 'Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94', ACM, pp. 258–264.
- Langlotz, T., Grubert, J. and Grasset, R. (2013), 'Augmented reality browsers: essential products or only gadgets?', *Commun. ACM* **56**(11), 34–36.
- Langlotz, T., Mooslechner, S., Zollmann, S., Degendorfer, C., Reitmayr, G. and Schmalstieg, D. (2012), 'Sketching up the world: in situ authoring for mobile augmented reality', *Personal and ubiquitous computing* **16**(6), 623–630.
- Langlotz, T., Nguyen, T., Schmalstieg, D. and Grasset, R. (2014), 'Next-generation augmented reality browsers: Rich, seamless, and adaptive', *Proceedings of the IEEE* **102**(2), 155–169.
- Langlotz, T., Regenbrecht, H., Zollmann, S. and Schmalstieg, D. (2013), Audio stickies: visually-guided spatial audio annotations on a mobile augmented reality platform, in 'Proceedings of the 25th Australian Computer-Human Interaction Conference on Augmentation, Application, Innovation, Collaboration - OzCHI '13', ACM, pp. 545–554.
- Langlotz, T., Zingerle, M., Grasset, R., Kaufmann, H. and Reitmayr, G. (2012), AR record&replay: situated compositing of video content in mobile augmented reality, in 'Proceedings of the 24th Australian Computer-Human Interaction Conference', ACM, pp. 318–326.
- Leal-Meléndrez, J. A., Altamirano-Robles, L. and Gonzalez, J. A. (2013), *Occlusion Handling in Video-Based Augmented Reality Using the Kinect Sensor for Indoor Registration*, Springer Berlin Heidelberg, pp. 447–454.

- Ledermann, F. and Schmalstieg, D. (2005), APRIL: a high-level framework for creating augmented reality presentations, in 'Virtual Reality, 2005. Proceedings. VR 2005. IEEE', IEEE, pp. 187–194.
- Lee, G. A., Yang, U., Kim, Y., Jo, D., Kim, K.-H., Kim, J. H. and Choi, J. S. (2009), Freeze-Set-Go interaction method for handheld mobile augmented reality environments, in 'Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology', ACM, pp. 143–146.
- Leung, R., McGrenere, J. and Graf, P. (2011), 'Age-related differences in the initial usability of mobile device icons', *Behaviour & Information Technology* **30**(5), 629–642.
- Lewis, C. (1982), *Using the "thinking-aloud" method in cognitive interface design*, IBM TJ Watson Research Center.
- Lewis, J. R. and Sauro, J. (2009), The factor structure of the system usability scale, in 'International conference on human centered design', Springer, pp. 94–103.
- Looser, J., Billingham, M. and Cockburn, A. (2004), Through the looking glass: the use of lenses as an interface tool for augmented reality interfaces, in 'Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia - GRAPHITE '04', ACM, pp. 204–211.
- MacIntyre, B., Gandy, M., Bolter, J., Dow, S. and Hannigan, B. (2003), DART: the designer's augmented reality toolkit, in 'Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on', pp. 329–330.
- MacIntyre, B., Gandy, M., Dow, S. and Bolter, J. D. (2004), DART: a toolkit for rapid design exploration of augmented reality experiences, in 'Proceedings of the 17th annual ACM symposium on User interface software and technology - UIST '04', ACM, pp. 197–206.
- MacLaurin, M. B. (2011), 'The design of Kodu: a tiny visual programming language for children on the Xbox 360', *SIGPLAN Not.* **46**(1), 241–246.
- Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. (2010), 'The Scratch programming language and environment', *ACM Transactions on Computing Education (TOCE)* **10**(4), 16.
- Mann, S. (2002), *The EyeTap Principle: Effectively Locating the Camera Inside the Eye as an Alternative to Wearable Camera Systems*, Wiley-IEEE Press, pp. 64–102.
- Mann, S. (2012), 'Through the glass, lightly [Viewpoint]', *IEEE Technology and Society Magazine* **31**(3), 10–14.
- Mann, S., Fung, J., Aimone, C., Sehgal, A. and Chen, D. (2005), 'Designing EyeTap digital eyeglasses for continuous lifelong capture and sharing of personal experiences', *Alt. Chi, Proc. CHI 2005* .
- Marneanu, I., Ebner, M. and Roessler, T. (2014), 'Evaluation of augmented reality frameworks for Android development', *iJIM* **8**(4), 37–44.

- Marzo, A. and Ardaiz, O. (2013), Collart: a tool for creating 3D photo collages using mobile augmented reality, in 'Proceedings of the 21st ACM international conference on Multimedia', ACM, pp. 585–588.
- Microsoft Corporation (2014), '3D Builder'. Accessed: November 2017.
URL: <https://www.microsoft.com/en-us/store/p/3d-builder/9wzdnrcrjf3t6>
- Microsoft Corporation (Research) (2017), 'Kodu Game Lab'. Accessed: July 2017.
URL: <https://www.kodugamelab.com>
- Milgram, P. and Kishino, F. (1994), 'A taxonomy of mixed reality visual displays', *IEICE TRANSACTIONS on Information and Systems* 77(12), 1321–1329. P
- Milgram, P., Takemura, H., Utsumi, A. and Kishino, F. (1994), Augmented reality: A class of displays on the reality-virtuality continuum, in 'Proceedings of Telemanipulator and Telepresence Technologies', Vol. 2351, pp. 282–292. P
- Mitchell, P. and Wilkinson, B. (2016), Periphery triggered menus for head mounted menu interface interactions, in 'Proceedings of the 28th Australian Conference on Computer-Human Interaction - OzCHI '16', ACM, pp. 30–33.
- Mogilev, D., Kiyokawa, K., Billingham, M. and Pair, J. (2002), AR Pad: An interface for face-to-face ar collaboration, in 'CHI '02 Extended Abstracts on Human Factors in Computing Systems', ACM, pp. 654–655.
- Mohring, M., Lessig, C. and Bimber, O. (2004), Video see-through AR on consumer cell-phones, in 'Third IEEE and ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 252–253.
- Mossel, A., Venditti, B. and Kaufmann, H. (2013a), 3DTouch and HOMER-S: intuitive manipulation techniques for one-handed handheld augmented reality, in 'Proceedings of the Virtual Reality International Conference: Laval Virtual', ACM, p. 12.
- Mossel, A., Venditti, B. and Kaufmann, H. (2013b), DrillSample: precise selection in dense handheld augmented reality environments, in 'Proceedings of the Virtual Reality International Conference: Laval Virtual', ACM, p. 10.
- Mulloni, A., Seichter, H. and Schmalstieg, D. (2011), Handheld augmented reality indoor navigation with activity-based instructions, in 'Proceedings of the 13th international conference on human computer interaction with mobile devices and services', ACM, pp. 211–220.
- Newman, J., Ingram, D. and Hopper, A. (2001), Augmented reality in a wide area sentient environment, in 'Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on', pp. 77–86. P
- Nielsen, J. (1994), Enhancing the explanatory power of usability heuristics, in 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '94, ACM, New York, NY, USA, pp. 152–158.
- Norman, D. A. (2013), *The design of everyday things: Revised and expanded edition*, Basic Books (AZ).

- Oda, O. and Feiner, S. (2012), 'Goblin XNA'. Accessed: July 2017.
 URL: <https://goblinxna.codeplex.com/>
- Papagiannakis, G., Singh, G. and Magnenat-Thalmann, N. (2008), 'A survey of mobile and wireless technologies for augmented reality systems', *Computer Animation and Virtual Worlds* **19**(1), 3–22.
- Park, H. M., Lee, S. H. and Choi, J. S. (2008), 'Wearable augmented reality system using gaze interaction', in '2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 175–176.
- Pereira, A., Miller, T., Huang, Y.-M., Odell, D. and Rempel, D. (2013), 'Holding a tablet computer with one hand: effect of tablet design features on biomechanics and subjective usability among users with small hands', *Ergonomics* **56**(9), 1363–1375.
- Piekarski, W. (2006), '3D modeling with the Tinmith mobile outdoor augmented reality system', *Computer Graphics and Applications, IEEE* **26**(1), 14–17.
- Piekarski, W., Avery, B., Thomas, B. H. and Malbezin, P. (2004), 'Integrated head and hand tracking for indoor and outdoor augmented reality', in 'Virtual Reality, 2004. Proceedings. IEEE', pp. 11–276.
- Piekarski, W. and Thomas, B. (2002), 'The Tinmith system: demonstrating new techniques for mobile augmented reality modelling', *Aust. Comput. Sci. Commun.* **24**(4), 61–70.
- Piekarski, W. and Thomas, B. H. (2004), 'Augmented reality working planes: A foundation for action and construction at a distance', in 'Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 162–171.
- Potter, R. L., Weldon, L. J. and Shneiderman, B. (1988), 'Improving the accuracy of touch screens: an experimental evaluation of three strategies', in 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 27–32.
- Prensky, M. (2001), 'Digital natives, digital immigrants part 1', *On the horizon* **9**(5), 1–6.
- Qualcomm Connected Experiences Inc. (2014), 'Vuforia'. Accessed: September 2014.
 URL: <https://www.vuforia.com>
- Rademacher, G. (2017), 'Railroad Diagram Generator 1.47.1494'. Accessed: July 2017.
 URL: <http://bottlecaps.de/rr/ui>
- Raskar, R., Welch, G. and Fuchs, H. (1999), 'Spatially augmented reality', in 'Proceedings of the International Workshop on Augmented Reality : Placing Artificial Objects in Real Scenes: Placing Artificial Objects in Real Scenes', IWAR '98, A. K. Peters, Ltd., Natick, MA, USA, pp. 63–72.
- Reisman, J. L., Davidson, P. L. and Han, J. Y. (2009), 'A screen-space formulation for 2D and 3D direct manipulation', in 'Proceedings of the 22nd annual ACM symposium on User interface software and technology', ACM, pp. 69–78.
- Rekimoto, J. (1995), 'The magnifying glass approach to augmented reality systems', in 'International Conference on Artificial Reality and Tele-Existence', Vol. 95, pp. 123–132.

- Rekimoto, J. and Nagao, K. (1995), The world through the computer: computer augmented interaction with real world environments, *in* 'Proceedings of the 8th annual ACM symposium on User interface and software technology - UIST '95', ACM, pp. 29–36.
- Roberto, R. A., Lima, J. P., Mota, R. C. and Teichrieb, V. (2016), Authoring tools for augmented reality: An analysis and classification of content design tools, *in* 'International Conference of Design, User Experience, and Usability', Springer, pp. 237–248.
- Rod, J., Li, C., Zhang, D. and Lee, H. (2016), Designing a 3D modelling tool for novice users, *in* 'Proceedings of the 28th Australian Conference on Computer-Human Interaction - OzCHI '16', ACM, pp. 140–144.
- Rolland, J. P. and Fuchs, H. (2000), 'Optical versus video see-through head-mounted displays in medical visualization', *Presence: Teleoperators and Virtual Environments* 9(3), 287–309.
- Rubin, J. and Chisnell, D. (2008), *Handbook of usability testing: how to plan, design and conduct effective tests*, John Wiley & Sons.
- Rubio, J. M. and Janecek, P. (2002), Floating pie menus: enhancing the functionality of contextual tools, *in* 'Proceedings of ACM UIST 2002 Conference Companion', pp. 39–40.
- Rumiński, D. and Walczak, K. (2013), Creation of interactive AR content on mobile devices, *in* 'Business Information Systems Workshops', Springer, pp. 258–269.
- Sambrooks, L. and Wilkinson, B. (2013), Comparison of gestural, touch, and mouse interaction with Fitts' law, *in* 'Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration', OzCHI '13, ACM, New York, NY, USA, pp. 119–122.
- Sambrooks, L. and Wilkinson, B. (2015), Handheld augmented reality: Does size matter?, *in* 'Proceedings of the 16th Australasian User Interface Conference (AUIC 2015)', Vol. 27, pp. 11–20.
- Sambrooks, L. and Wilkinson, B. (2016), Designing HARATIO: A novice AR authoring tool, *in* 'Proceedings of the 28th Australian Conference on Computer-Human Interaction', OzCHI '16, ACM, New York, NY, USA, pp. 175–179.
- Samp, K. and Decker, S. (2010), Supporting menu design with radial layouts, *in* 'Proceedings of the International Conference on Advanced Visual Interfaces', ACM, pp. 155–162.
- Samp, K. and Decker, S. (2011), 'Visual search in radial menus', *Human-Computer Interaction* 26(3), 248–255.
- Santos, M. E. C., Polvi, J., Taketomi, T., Yamamoto, G., Sandor, C. and Kato, H. (2015), 'Toward standard usability questionnaires for handheld augmented reality', *Computer Graphics and Applications, IEEE* 35(5), 66–75.
- Santos, M. E. C., Taketomi, T., Sandor, C., Polvi, J., Yamamoto, G. and Kato, H. (2014), A usability scale for handheld augmented reality, *in* 'Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology', ACM, pp. 167–176.

- Sauro, J. (2011), 'Measuring user interface disasters'. Accessed: February 2017.
 URL: <http://www.measuringu.com/blog/measuring-confidence.php>
- Sauro, J. (2012), '10 things to know about the single ease question (SEQ)'. Accessed: May 2017.
 URL: <https://measuringu.com/seq10/>
- Sauro, J. and Dumas, J. S. (2009), Comparison of three one-question, post-task usability questionnaires, in 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', ACM, pp. 1599–1608.
- Sauro, J. and Lewis, J. R. (2010), Average task times in usability tests: what to report?, in 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', ACM, pp. 2347–2350.
- Sauro, J. and Lewis, J. R. (2012), *Quantifying the User Experience: Practical Statistics for User Research*, 1st edn, Morgan Kaufmann Publishers Inc.
- Schall, G., Mendez, E., Kruijff, E., Veas, E., Junghanns, S., Reitingen, B. and Schmalstieg, D. (2009), 'Handheld augmented reality for underground infrastructure visualization', *Personal and Ubiquitous Computing* **13**(4), 281–291.
- Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavári, Z., Encarnaçao, L. M., Gervautz, M. and Purgathofer, W. (2002), 'The Studierstube augmented reality project', *Presence: Teleoperators & Virtual Environments* **11**(1), 33–54.
- Schmalstieg, D., Langlotz, T. and Billinghurst, M. (2011), Augmented reality 2.0, in 'Virtual realities', Springer, pp. 13–37.
- Schmalstieg, D. and Wagner, D. (2007), Experiences with handheld augmented reality, in 'Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on', pp. 3–18.
- Seichter, H., Looser, J. and Billinghurst, M. (2008), ComposAR: An intuitive tool for authoring AR applications, in 'Proceedings of the 7th IEEE/ACM international symposium on mixed and augmented reality', IEEE Computer Society, pp. 177–178.
- Selwyn, N. (2009), The digital native – myth and reality, in 'Aslib Proceedings', Vol. 61, Emerald Group Publishing Limited, pp. 364–379.
- Shneiderman, B. (1983), 'Direct manipulation – a step beyond programming languages', *Computer* **16**(8), 57–69. ISI Document Delivery No.: RB386 Times Cited: 89 Cited Reference Count: 31 Shneiderman, b Ieee computer soc Los alamos.
- Singh, M. and Singh, M. P. (2013), 'Augmented reality interfaces', *Internet Computing, IEEE* **17**(6), 66–70.
- Somberg, B. L. (1987), 'A comparison of rule-based and positionally constant arrangements of computer menu items', *ACM SIGCHI Bulletin* **18**(4), 255–260.
- Stolee, K. T. (2010), 'Kodu language and grammar specification', *Microsoft Research whitepaper*, Retrieved September 1, 4–6.

- Stolee, K. T. and Fristoe, T. (2011), Expressing computer science concepts through Kodu game lab, in 'Proceedings of the 42nd ACM technical symposium on Computer science education', ACM, pp. 99–104.
- Sugihara, T. and Miyasato, T. (1998), A lightweight 3-D HMD with accommodative compensation, in 'SID Symposium Digest of Technical Papers', Vol. 29, Wiley Online Library, pp. 927–930.
- Sutherland, I. E. (1968), A head-mounted three dimensional display, in 'Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)', ACM, pp. 757–764.
- Tang, J. K. T., Duong, T. Y. A., Ng, Y. W. and Luk, H. K. (2015), Learning to create 3D models via an augmented reality smartphone interface, in 'Teaching, Assessment, and Learning for Engineering (TALE), 2015 IEEE International Conference on', pp. 236–241.
- Tapia, M. A. and Kurtenbach, G. (1995), Some design refinements and principles on the appearance and behavior of marking menus, in 'Proceedings of the 8th annual ACM symposium on User interface and software technology', ACM, pp. 189–195.
- The LEGO Group (2017), 'LEGO Mindstorms'. Accessed: July 2017.
URL: <http://mindstorms.lego.com>
- Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P. and Piekarski, W. (2002), 'First person indoor/outdoor augmented reality application: ARQuake', *Personal Ubiquitous Comput.* **6**(1), 75–86.
- Tokusho, Y. and Feiner, S. (2009), Prototyping an outdoor mobile augmented reality street view application, in 'Proceedings of ISMAR Workshop on Outdoor Mixed and Augmented Reality', Vol. 2, Citeseer.
- Unity Technologies SF (2014a), 'Unity 4'. Accessed: April 2014.
URL: <https://unity3d.com/unity>
- Unity Technologies SF (2014b), 'Unity Manual (Version 4.6): ShaderLab syntax: SubShader Tags'. Accessed: June 2017.
URL: <https://docs.unity3d.com/460/Documentation/Manual/SL-SubshaderTags.html>
- Veas, E. E. and Kruijff, E. (2010), Handheld devices for mobile augmented reality, in 'Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia', ACM, p. 3.
- Veas, E. and Kruijff, E. (2008), Vesp'R: design and evaluation of a handheld ar device, in 'Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 43–52.
- Ventura, J., Arth, C., Reitmayr, G. and Schmalstieg, D. (2014), 'Global localization from monocular SLAM on a mobile phone', *IEEE Transactions on Visualization and Computer Graphics* **20**(4), 531–539.
- Vincent, T., Nigay, L. and Kurata, T. (2013), *Precise Pointing Techniques for Handheld Augmented Reality*, Vol. 8117 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, book section 9, pp. 122–139.

- Vogel, D. and Baudisch, P. (2007), Shift: a technique for operating pen-based interfaces using touch, in 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 657–666.
- Wagner, D. (2007), Handheld augmented reality, Thesis.
- Wagner, D., Langlotz, T. and Schmalstieg, D. (2008), Robust and unobtrusive marker tracking on mobile phones, in '2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 121–124.
- Wagner, D., Pintaric, T., Ledermann, F. and Schmalstieg, D. (2005), *Towards Massively Multi-user Augmented Reality on Handheld Devices*, Vol. 3468 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, book section 13, pp. 208–219.
- Wagner, D., Pintaric, T. and Schmalstieg, D. (2004), The invisible train: a collaborative handheld augmented reality demonstrator, in 'ACM SIGGRAPH 2004 Emerging technologies on - SIGGRAPH '04', ACM, p. 12.
- Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T. and Schmalstieg, D. (2008), Pose tracking from natural features on mobile phones, in 'Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 125–134.
- Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T. and Schmalstieg, D. (2010), 'Real-time detection and tracking for augmented reality on mobile phones', *Visualization and Computer Graphics, IEEE Transactions on* **16**(3), 355–368.
- Wagner, D. and Schmalstieg, D. (2003), First steps towards handheld augmented reality, in 'Proceedings of the 7th International Symposium on Wearable Computers (ISWC'2003)', pp. 127–137.
- Wagner, D. and Schmalstieg, D. (2006), Handheld augmented reality displays, in 'Virtual Reality Conference, 2006', pp. 321–321.
- Wagner, D. and Schmalstieg, D. (2007), ARToolKitPlus for pose tracking on mobile devices, in 'Proceedings of 12th Computer Vision Winter Workshop'.
- Wang, Y., Langlotz, T., Billinghamurst, M. and Bell, T. (2009), An authoring tool for mobile phone AR environments, in 'Proceedings of New Zealand Computer Science Research Student Conference', Vol. 9, pp. 1–4.
- Wellner, P. (1993), 'Interacting with paper on the DigitalDesk', *Commun. ACM* **36**(7), 87–96.
- Wiedenbeck, S. (1999), 'The use of icons and labels in an end user application program: an empirical study of learning and retention', *Behaviour & Information Technology* **18**(2), 68–82.
- Wilkinson, B. and Calder, P. (2006), Augmented reality for the real world, in 'Computer Graphics, Imaging and Visualisation, 2006 International Conference on', pp. 452–457.
- Wither, J., DiVerdi, S. and Höllerer, T. (2007), Evaluating display types for ar selection and annotation, in 'Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 1–4.

- Wloka, M. M. and Anderson, B. G. (1995), Resolving occlusion in augmented reality, in 'Proceedings of the 1995 symposium on Interactive 3D graphics', ACM, pp. 5–12.
- Wozniowski, M. and Warne, P. (2011), Towards in situ authoring of augmented reality content, in 'Proceedings ISMAR'.
- Yang, Y., Shim, J., Chae, S. and Han, T.-D. (2016a), Interactive augmented reality authoring system using mobile device as input method, in 'Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on', IEEE, pp. 001429–001432.
- Yang, Y., Shim, J., Chae, S. and Han, T.-D. (2016b), Mobile augmented reality authoring tool, in '2016 IEEE Tenth International Conference on Semantic Computing (ICSC)', IEEE, pp. 358–361.
- Yoo, H. and Lee, J. (2015), 'Mobile augmented reality authoring system with 3D modeling and lighting estimation', *Appl. Math* 9(2L), 553–562.
- Zauner, J. and Haller, M. (2004), Authoring of mixed reality applications including multi-marker calibration for mobile devices, in '10th Eurographics Symposium on Virtual Environments, EGVE', pp. 87–90.
- Zdziebło, V. (2015), 'Unlimited Screen Recorder 1.10'. Accessed: November 2017.
URL: <https://play.google.com/store/apps/details?id=org.mistygames.screenrecord>
- Zhou, F., Duh, H. B.-L. and Billinghamst, M. (2008), Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR, in 'Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality', IEEE Computer Society, pp. 193–202.