3DUITK: An Opensource Toolkit for Thirty Years of Three-Dimensional Interaction Research

Kieran W. May*
University of South Australia
Mawson Lakes

Ian Hanan†
University of South Australia
Mawson Lakes

Andrew Cunningham‡
University of South Australia
Mawson Lakes

Bruce H. Thomas§
University of South Australia
Mawson Lakes

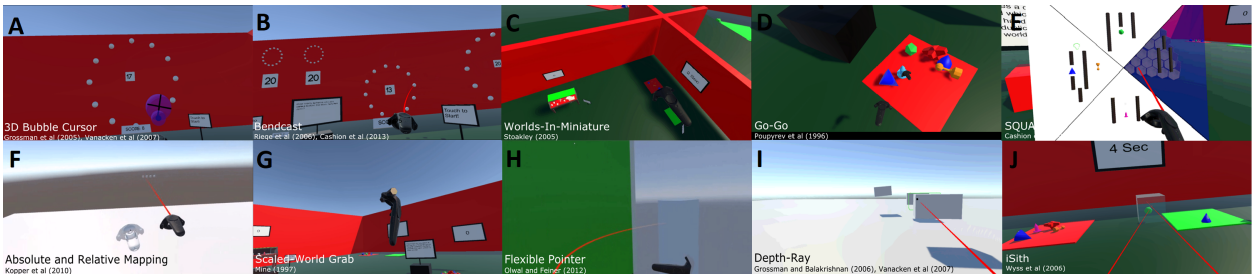

Figure 1: Ten of the twenty-five interactions techniques supported by the 3D User Interaction Toolkit: (A) 3D Bubble Cursor, (B) Bendcast, (C) Worlds In Miniature, (D) Go-Go, (E) SQUAD, (F) Absolute and Relative Mapping, (G) Scaled-World Grab, (H) Flexible Pointer, (I) Depth Ray, (J) iSith.

Abstract

We present the 3D User Interaction Toolkit (3DUITK), an open-source toolkit consisting of 25 Virtual Reality (VR) interaction techniques proposed from thirty years of prior research. Although interaction has been a topic heavily explored in previous papers, much of the research is yet to be re-implemented into an easy to use toolkit. Existing open-source VR interaction solutions provide basic functionality with few available techniques to choose from, and there has yet to be an open-source immersive user interaction toolkit to date. In this paper, we provide an overview of 3DUITK which includes the features, related works, design decisions, example applications, limitations, and future directions. While the focus of this work is on the VR context, 3DUITK has been employed in spatial augmented reality and mobile augmented reality applications.

Index Terms: Human Computer Interaction—3D User Interfaces—Interaction Technique—Toolkit; Virtual Reality—HTC Vive—Oculus Rift—Unity;

1 Introduction

In this paper we introduce 3DUITK, an open-source [1] 3D user interaction toolkit consisting of twenty-five immersive interaction techniques proposed from thirty years of prior research. Three-dimensional interaction is one of the core design decisions during the development of an interactive Virtual Reality (VR) application; the ideal technique for interacting with a virtual object is highly dependent on the particular tasks the user is performing. As a toolkit, 3DUITK provides developers a set of interaction techniques to address a variety of task conditions, and provides researchers with baseline implementations of classic techniques for further investigations.

*kieran.may@mymail.unisa.edu.au
†e-mail: hanis001@mymail.unisa.edu.au
‡e-mail: Andrew.Cunningham@unisa.edu.au
§e-mail: Bruce.Thomas@unisa.edu.au

[1] https://github.com/WearableComputerLab/VRInteractionToolkit

The two most prevalent interaction techniques used in modern VR applications are ray-casting [24] — where a virtual-ray is cast from the user's controller to select objects — and simple virtual-hand [15] — a one-to-one mapping between the user's physical hand and a virtual hand providing a direct form of interaction. Although these techniques perform well in simple tasks, limitations arise when changing task conditions. For example, virtual-hand is limited to only interacting with objects within the user's physical reach; ray-casting addresses the reach issue but performs poorly when presented with small, distant, or densely packed objects.

To address these issues, researchers over the past thirty-years have explored alternative interaction solutions, proposing a variety of techniques optimized for particular tasks (see Figure 1). However, many of the proposed techniques do not have a modern implementation, while some of the older techniques have only a partial description and require interpretation to be implemented on modern hardware. Existing open-source VR toolkits [1,2] compatible with the Unity 3D game engine provide limited interaction functionality and primarily consist of ray-casting and simple virtual-hand techniques.

Our goals from this work is to provide the research community with an open-source toolkit dedicated to 3D user interaction techniques capable of addressing a wide range of task requirements. As such, we believe LaViola et al.'s [19] work is definitive in describing the state of the art for 3D user interactions research. Based on LaViola et al. [19, Chapter 7], we categorize interaction techniques into two subcategories: 1) **selection** — a method of selecting an object without modifying or making any physical changes to that object, and 2) **manipulation** — a method of modifying the *position*, *rotation*, or *scale* of an object. Within these categories, we have identified twenty-five techniques suitable for implementation in 3DUITK.

Our implementations of these techniques follow a general approach of addressing missing design details and considerations from original papers. The modern implementations we have built for 3DUITK runs on the HTC Vive and Oculus Rift hardware using the Unity 3D game engine.

The specific contributions of this paper are:

- An **open source toolkit** for immersive interaction.
- A **reference implementation** of twenty-five interaction techniques that address missing design details.
- A **design for a common interface** for the selection and manipulation techniques.

- A **comprehensive testing scene** for testing interaction techniques under a variety of conditions (see 4.3).
- A flexible implementation to support **different display technologies**.

2 Related work

Prior to the current wave of modern game engines and VR platforms, general VR frameworks such as Alice [9] and VR Juggler [3] have included some support for user interaction. In recent times, two VR frameworks for modern platforms provide basic VR interaction solutions. The Virtual Reality Toolkit (VRTK)[2] is an open-source toolkit designed to provide developers with a building-block framework for the development of VR applications within Unity. The VRTK framework is comprised of a variety of scripts relating to locomotion, 2D and 3D user-interface interaction, body physics, and interactive object functionality (i.e. levers, doors, drawers etc.). In regards to user interaction, VRTK provides support for either the virtual hand or ray-casting techniques.

Similarly, Newton VR[3] is a Unity 3D toolkit focused on realistic physics-integration in VR, providing pick, drop, throw, and hold functionality using the virtual hand interaction technique. Whilst both VRTK and Newton VR provide user-interaction techniques, each toolkit's focus was on other aspects of VR and rely on basic user-interaction solutions to handle the selection and manipulation task requirements. Due to the lack of user interaction support provided from these toolkits, many applications built upon these frameworks are severely disadvantaged by the limited interaction solutions. We see 3DUITK as being complementary to these aforementioned toolkits, and encourage the use of 3DUITK with other open-source toolkits and libraries. This allows developers and researchers to maintain the core functionality of the mentioned toolkits whilst gaining access to a plethora of user interaction solutions to choose from.

3 Interaction Techniques

The twenty-five techniques implemented in 3DUITK (Table 1) are motivated by LaViola et al.'s thorough review of 3D interaction [19, Chapter 7]. LaViola et al. identify four canonical tasks for 3D interaction: *selection, positioning, rotation, and scaling*. These tasks are further categorized into *selection*, and *manipulation* (which encompasses the position, rotation, and scaling tasks). Additionally, the authors organize interaction techniques into six metaphor categories: *Grasping, Pointing, Surface, Indirect, Bimanual, and Hybrid*.

This section provides an overview of these five metaphor categories as they provide a classification of the twenty-five implementations presented in 3DUITK. Due to the hardware restrictions outlined in the limitations (Section 6), the surface metaphor is excluded from the toolkit as there is no commonly accepted multi-touch surface for VR. Table 1 highlights the interaction techniques implemented in 3DUITK categorized by their corresponding metaphor.

3.1 Grasping

Hand-grasping techniques provide a natural, intuitive form of 3D interaction by allowing users to directly interact with objects using their virtual hands. Due to the limitations associated with interacting with objects positioned beyond the physical reach of the user a number of grasping enhancements are proposed [12, 13, 27, 34].

3.2 Pointing

Traditional pointing techniques (i.e. ray-casting) still remains the most widely-used form of selection-based interaction in modern VR applications. The pointing metaphor describes techniques which provide object interaction beyond the users physical reach through the use of pointing mechanisms. Many pointing techniques implemented in 3DUITK utilize non-isomorphic ("Magic") pointing tools to act as a visual guidance providing input feedback to the user.

3.3 Indirect

Indirect techniques refer to performing selection or manipulation tasks without directly interacting with the intended object. This process allows users to seamlessly interact with multiple objects at a time without having to travel. One of the most notable interaction techniques in human-computer interaction *Worlds In Miniature (WIM)* [33] demonstrates the potential benefits of indirect interaction.

3.4 Bimanual

The bimanual metaphor encompasses all interaction technique that require the use of both hands when performing an interaction task. Multi-hand interaction techniques are generally associated with an increased number of degrees of freedom (DOF) due to having additional tracked input mechanisms (e.g. two 6DOF Vive controllers). Naturally bimanual techniques result in a more difficult form of interaction due to the increased DOF. However, once mastered bimanual techniques can be significantly advantageous by providing a much more precise form of manipulation in comparison to other metaphor categories discussed.

3.5 Hybrid

Hybrid techniques combine two or more interaction techniques from varying metaphor categories into a single technique. The hybrid structure aims to capture the best qualities from multiple interaction techniques and merge them together to meet a specific set of task requirements. Many manipulation techniques listed in our toolkit naturally rely on a hybrid structure as they require an initial selection technique to firstly identify the object that will be manipulated.

4 Design Overview

The design considerations we established stemmed from our initial goal of providing developers with a feasible interaction toolkit. In order to achieve this goal, we identified the following set of design requirements:

DR1—Adhere to the original design when possible: Implementing some of the user interaction techniques presented a variety of challenges, either from a lack of detailed information regarding the technique, insufficient algorithmic information on the design of the technique, or differences in the functionality of the original hardware and software compared to modern hardware and the Unity software. The hardware and software limitations mentioned were primarily due to techniques being initially designed on significantly older, less-powerful VR hardware. Instead of addressing these issues by taking advantage of features of the modern VR systems, we made the design decision to strictly adhere to the original design of the user interaction techniques where possible.

DR2—Multi-environment support: Implementations are designed to provide support for the HTC Vive and Oculus Rift VR platforms.

DR3—Ease of use: Many of the core design aspects associated with the framework aim to provide developers with a user-friendly approach for customizing and managing implementations. These customizations consist of managing techniques directly from the Unity 3D inspector interface (Section 4.2.1) and event systems (Section 4.2.2). Furthermore, a drag and drop design utilizing Unity 3D prefabs was adopted to simplify the process of integrating implementations into pre-existing scenes (Section 4.1).

DR4—Limited Dependencies: Technique implementations in 3DUITK adhere to a structurally independent design-flow. This means all components associated with a technique (e.g. Materials,

[2] https://vrtoolkit.readme.io

[3] http://www.newtonvr.com/

Table 1: Interaction techniques implemented in 3DUITK

Grasping		Pointing			Bimanual			Hybrid		
Hand-Grasping	Enhanced-Grasping	Vector-Based Pointing	Volume-Based Pointing	Enhanced Pointing	Proxy (Indirect)	Symmetric	Asymetric	Integrated	Muliple-Object Selection	Progressive Refinement
Simple Virtual-Hand [15]	3D Bubble Cursor [13, 34]	Ray-Casting [24]	Flashlight [20]	Bendcast [7, 31]	Worlds In Miniature [33]	Spindle [23, 32]	Spindle + Wheel [8]	Homer [4]	Serial selection [22]	SQUAD [6]
Go-Go [29]	PRISM [12]	Fishing Reel [5]	Aperture Selection [11]	Depth Ray [14, 21]		iSith [35]	Flexible pointer [26]	Scaled-world grab [25]		EXPAND [16]
	Hook [27]	Image-Plane Pointing [28]	Sphere-casting [17]	Absolute and relative mapping [18]						Double Bubble [10]

classes, prefabs) are designed to not be accessible by other implementations. The motivation behind minimizing dependencies is to allow interaction techniques to act as its own separate implementation. Therefore, modifications made to an interaction technique will not effect the functionality or design of the other twenty-four implementations in the toolkit.

DR5—Reliability: Since the toolkit is released as a publicly open-sourced framework, minimizing code errors was a crucial consideration. A paired-programming approach was adopted to mitigate code discrepancies throughout development. Additional strategies to support reliability consisted of developing testing scenes to test the functionality of the interaction techniques under a number of task conditions (Section 4.3). The performance of heavy-algorithmic based implementations were evaluated and analyzed with a custom made stress-testing scene (Section 4.3.1).

4.1 Prefabs

The Unity 3D prefab system allows for reusable game object components to be stored within a projects directory. Using this concept, 3DUITK presents each interaction technique as a prefab that can be implemented simply by dragging the prefab from the project directory into a scenes hierarchy. Once a technique is dropped into a projects hierarchy, the prefab will autonomously detect the controllers and game objects within the scene.

4.2 Inspector Design

Although the code-base for the individual interaction techniques are independent, the implementations still follow a broad design goal to improve the usability of the toolkit. These design goals were motivated by the natural built-in workflow of the Unity 3D game engine. As a result, the framework of 3DUITK is designed as an inspector-driven design which allows modifications to be made directly through the Unity inspector using parameters and events.

4.2.1 Parameters

The built-in inspector parameter architecture of Unity 3D provides developers with the capability of modifying variables directly through the Unity inspector window rather than having to manually modify code of scripts. In order to ensure 3DUITK strictly abides by this structure, a number of variables describing the properties associated with implementations are made publicly visible to the inspector. This is demonstrated in Figure 2, where the interaction technique *Worlds In Miniature* is customized to a 1:20 scale, specific game objects are ignored, and the rotational tilt speed of the miniature world is specified.

Each implementation provides support for the following three interaction settings: *Pure-selection*, *Manipulation (Movement Only)*, and *Manipulation (Menu)*. The purpose of these layers is designed for developers to configure interaction techniques to follow a specific form of interaction. For example, applying the pure-selection

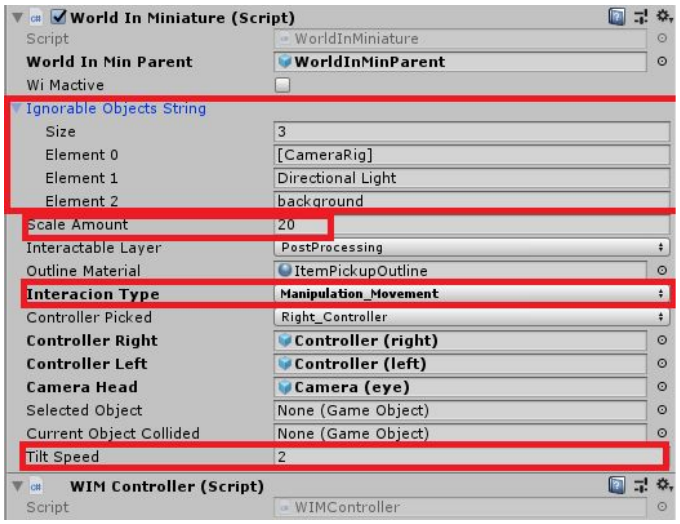

Figure 2: World-In-Miniature Inspector Parameters

constraint will restrict all implementations to performing selection only tasks. Upon selection, a reference of the selected object is stored within the inspector and no additional manipulation occurs upon the selected object unless instructed to do so by the event-system. When overriding a manipulation technique by applying the pure-selection constraint, the manipulation aspect of the technique essentially becomes obsolete which essentially results in the technique now becoming a selection technique. By default, manipulation techniques are configured to the Manipulation (Movement Only) constraint which describes the natural functionality of manipulation techniques. Upon selection, the position and rotation of an object are modified using the designated manipulation technique. Lastly, the Manipulation (Menu) constraint is designed to act as an enhancement for selection techniques by providing additional manipulation options upon selection. The menu is comprised of four manipulation tools: movement, scaling, delete, and colour picker. This menu serves as an exemplar for a developer to modify as required.

4.2.2 Event Systems

Similar to the reasons outlined in Section 4.2.1, an event-system architecture was developed based on the Unity events system providing a user-friendly inspector-driven customization of techniques (see Figure 3). Our event-system is comprised of four core events: `SelectedObject`: Events which occur upon object selection; `DroppedObject`: Events which occur when an object is unselected; `Hovered`: Events which occur when a technique intersects the object; and `UnHovered`: Events which occur when a technique

is no longer intersecting an object. These four events were built as a guideline for developers to incorporate manipulation functionality into selection based interaction techniques.

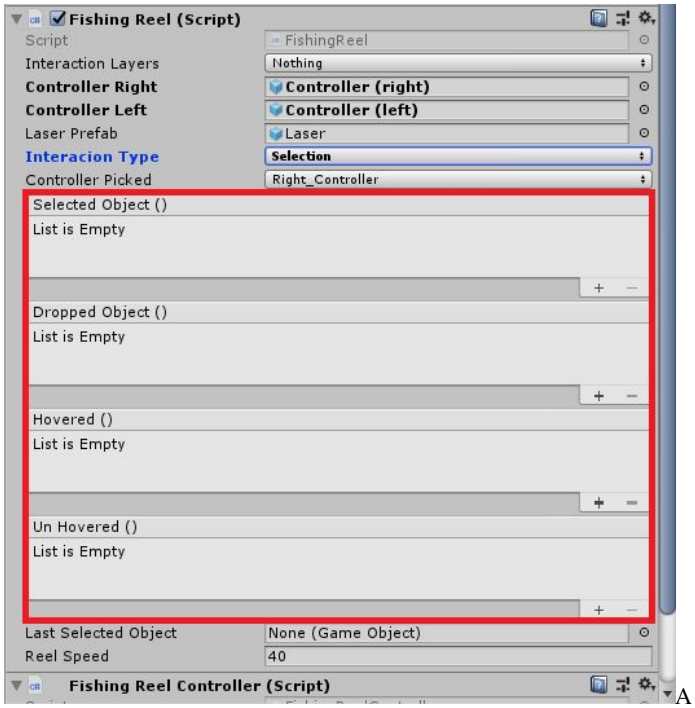

Figure 3: The Event-driven user interface of 3DUITK running in Unity 3D.

4.3 Testing Scene

Each interaction technique has two testing scenes to demonstrate and test the functionality of each implementation. The *Basic Testing Scene* consists of basic 3D primitive objects and is primarily designed to act as both a basic-training and development area to understand and test different aspects of interaction techniques. The *Major Testing Scene* incorporates a much more comprehensive testing structure. The major testing scene is designed to allow developers to simulate the functionality of interaction techniques under a wide-range of conditions. A game-like structure is adopted utilizing a high-score system to provide developers with a competitive approach to compare interaction techniques and determine their strengths and weaknesses for various task conditions. Each task presented in the major testing scene is designed to evaluate one of the following three core tasks:

- **Selection Tasks:** Tasks which are designed to compare pure-selection techniques under a variety of selection conditions. These conditions differ based on the shape, height, size of the game objects, and the view-point perspective of the user.
- **Manipulation Tasks:** The manipulation tasks primarily consist of selecting an object and moving it from one position to another. Similar to the selection tasks, each manipulation task also consists of variating conditions.
- **Docker Tasks:** The purpose of the docker-task is to compare the rotational and fine movement aspects of manipulation techniques. The objective of the task is to select and move an object to fit into a replicated silhouette of itself.

4.3.1 Stress Testing

The capability and performance of 3DUITK was measured through the use of a custom-made stress testing scene designed to test heavy-algorithmic based implementations. The implemented stress-test scene consisted of an initially empty environment where thousands of primitive game objects are gradually spawned over a specified time interval and at random positions within a 100×100 meter area around the user. The results from these tests validated the performance of our toolkit as implementations were able to maintain a steady 90 FPS on a standard Intel i7 with an Nvidia GTX 970 graphics card.

5 Example Applications

In addition to the major testing scenes discussed in Section 4.3, the functionality of 3DUITK integration with pre-existing projects was tested with a variety of free virtual environments downloaded from the Unity Asset Store. The outcome of these tests revealed implementations were capable of seamlessly integrating with pre-existing virtual environments under two conditions:

1. Game objects in the VE must *contain a collider* in order to be recognized by the software.
2. The VE is not already comprised of a *high-level of user-interaction* in order to avoid potential clash discrepancies.

Furthermore, we built two example applications running on significantly different display platforms to demonstrate the adaptability of our toolkit: Spatial Augmented Reality and Mobile Augmented reality.

5.1 Spatial Sugmented Reality (SAR) Application

We explored the implementation of 3DUITK interaction techniques inside a projector-based SAR [30] environment with HTC Vive tracking (see Figure 5). The toolkit essentially proved to be very adaptable by successfully integrating to the SAR environment without requiring any modifications. Although some interaction techniques are not suited to the 2D projected nature of SAR, we found Bendcast, 3D Bubble Cursor, Image-Plane Interaction, Flashlight, Aperture, Ray-casting, Sphere-Casting, and Serial selection to work effectively in this environment.

5.2 Mobile augmented reality (MAR) Application

Next we tested 3DUITK techniques inside a marker-based mobile augmented reality (MAR) android application (Figure 5). Although we only assessed four interaction implementations: Ray-casting, 3D Bubble Cursor, Bendcast, and Image-Plane interaction. The tests revealed positive results as all four implementations provided unique and effective ways of handling MAR user-interaction. Unlike the SAR application, some modifications were required to make 3DUITK compatible with an android device. These modifications included replacing the SteamVR input system with a tangible android-based input system. Although this conversion was a relatively simple process, it is evident that future implementations of 3DUITK on hardware with a lack of SteamVR input-support will require additional code to handle the hardware's input requirements. Furthermore, due to the smart-phone essentially acting as both the HMD and controller, the interaction techniques that required either two-handed bimanual interaction or multi-input event handling would not be feasible for a MAR application.

Although extensive research was not conducted in regards to evaluating 3DUITK with additional hardware displays. Our preliminary tests revealed 3DUITK, a toolkit built specifically for an HMD VR device was capable of adapting to with two forms of AR with considerably different hardware platforms. These platforms consisted of a projector-based SAR display, which projected virtual information onto a 2D surface, and an android-based MAR display, which augmented 3D virtual information over the physical world.

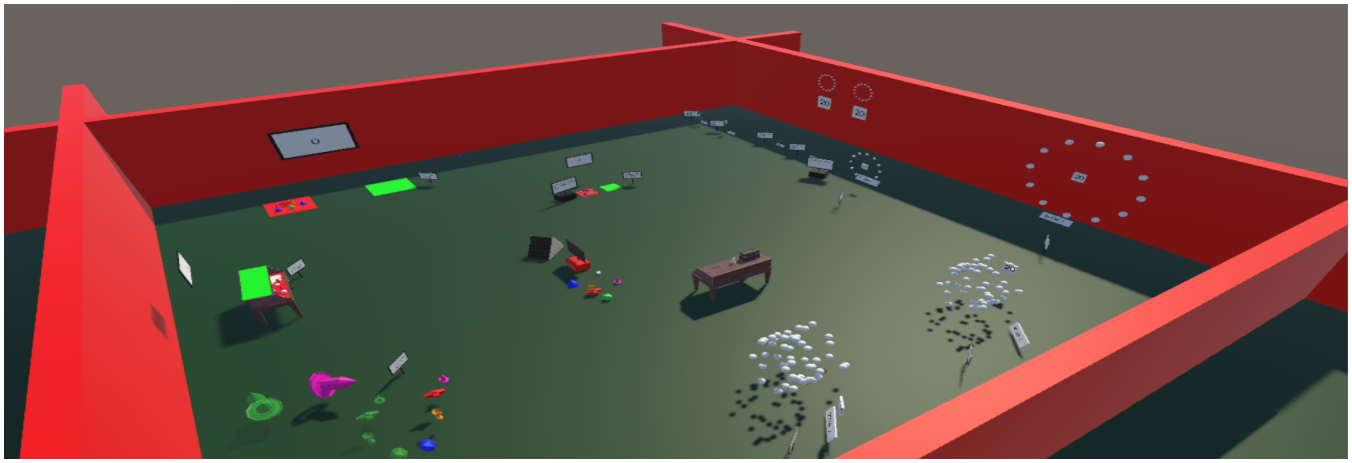

Figure 4: The Major Testing Scene. Users can move around the testing scene and perform a variety of tasks that test the limit of each technique.

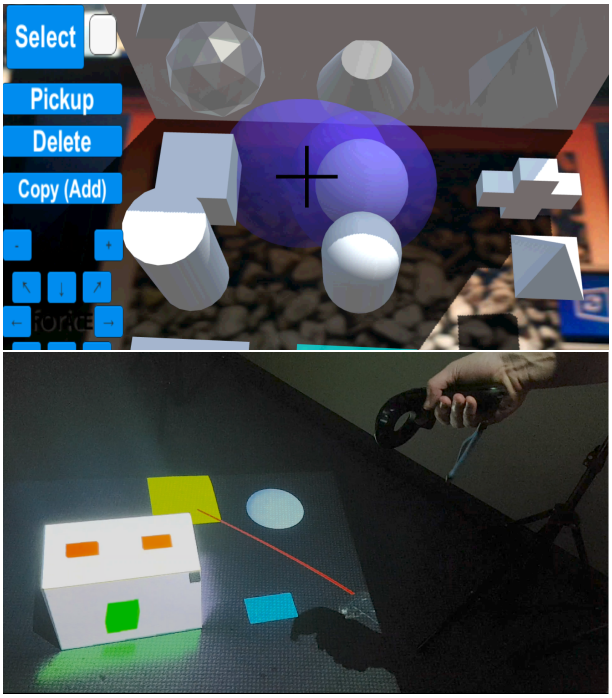

Figure 5: Two demonstrations of the 3DUITK adapted to AR hardware. (top) the 3D Bubble Cursor operating on a mobile AR device with marker-based tracking. (bottom) Bendcast operating inside a project-based SAR environment.

6 Limitations

Although we validated that 3DUITK can be integrated into several pre-existing projects, we presume potential discrepancies may arise if integrated into a project that already contained a high-degree of customized user interaction. In this case, customization through the Inspector is most likely not going to be feasible to address these discrepancies, and the developer may be required to make additional changes to the code.

Hardware limitations: The hardware limitations associated with the standard HTC Vive and Oculus Rift VR kits resulted in nine interaction techniques summarized by LaViola et al. [19] to be excluded from 3DUITK. These limitations boiled down to two primary causes:

1. *Additional hardware requirements* resulted in seven interaction techniques to be excluded; one requiring a smart-phone, six requiring multi-touch surfaces.
2. *Hand and finger gesture tracking*, which accounted for the remaining two interaction techniques to be excluded.

SteamVR limitations: All twenty-five implementations in 3DUITK are integrated with the SteamVR Library, which provides controller input and camera rig functionality for the HTC Vive and Oculus Rift HMDs. Having 3DUITK reliant on the SteamVR library can pose potentially problematic contingencies as a SteamVR software update could cause discrepancies to occur in implementations. To address this potential issue, a SteamVR version compatible with 3DUITK is packaged with the toolkit which provides a functional version of the toolkit at all times.

7 Future Directions

Although initial implementations of 3DUITK were designed specifically for Virtual Reality HMDs (HTC Vive and Oculus Rift). Further experiments (Section 5) demonstrated that 3DUITK could easily be adapted to integrate with other hardware displays. This is a major potential future direction as 3DUITK can be applied to related immersive fields such as Augmented and Mixed Reality. As 3DUITK is to be released open-source project, we envision additional interaction techniques to be added to the toolkit.

8 Conclusion

This paper presents an adaptable open-source 3D User-interaction Virtual Reality toolkit consisting of twenty-five interaction techniques implementations designed from thirty years of 3D interaction research. A summary of the toolkit is provided describing details regarding the design of the toolkit, categories of the interaction techniques, related works, limitations, and potential future directions. [4] Our goal for this work is to provide developers and researchers with an interaction toolkit they can utilise and build upon. Developers will be able to levarge a variety of user interactions designed to meet a wide range of requirements and user-experience goals. Researchers will be able to build upon the toolkit to define new interaction methods and test the methods in a variety of common testing scenes. Our hope is that this work will help accelerate both research and development in 3D user interaction.

[4] A demonstration of the twenty-five interaction technique implementations can be found at: https://youtu.be/nGMC6kkdjg8

REFERENCES

[1] https://vrtoolkit.readme.io/docs.

[2] http://www.newtonvr.com/.

[3] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *Proceedings IEEE Virtual Reality 2001*, pp. 89–96. IEEE, 2001.

[4] D. A. Bowman and L. F. Hodges. An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. *Proceedings of the 1997 ACM Symposium on Interactive 3D Graphics (I3D 97)*, pp. 35–38, 1997. doi: 10.1145/253284.253301

[5] D. A. Bowman and L. F. Hodges. An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. *Proceedings of the 1997 ACM Symposium on Interactive 3D Graphics (I3D 97)*, pp. 35–38, 1997.

[6] J. Cashion, C. Wingrave, and J. J. L. Jr. Dense and Dynamic 3D Selection for Game-Based Virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, pp. 634–642, 2012. doi: 10.1109/TVCG.2012.40

[7] J. Cashion, C. Wingrave, and J. J. LaViola. Optimal 3D Selection Technique Assignment using Real-Time Contextual Analysis. *Proceedings of the 2013 IEEE Symposium on 3D User Interfaces (3DUI 13)*, pp. 107–110, 2013. doi: 10.1109/3DUI.2013.6550205

[8] I. Cho and Z. Wartell. Evaluation of a Bimanual Simultaneous 7DOF Interaction Technique in Virtual Environments. *Proceedings of the 2015 IEEE Symposium on 3D User Interfaces (3DUI 15)*, pp. 133–136, 2015. doi: 10.1109/3DUI.2015.7131738

[9] M. Conway, R. Pausch, R. Gossweiler, and T. Burnette. Alice: a rapid prototyping system for building virtual environments. In *CHI Conference Companion*, pp. 295–296. Citeseer, 1994.

[10] F. B. de Araujo e Silva. *Increasing Selection Accuracy and Speed through Progressive Refinement*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2015. doi: 10919/56658

[11] A. Forsberg, K. Herndon, and R. Zeleznik. Aperture Based Selection for Immersive Virtual Environments. *Proceedings of the 1996 ACM Symposium on User Interface Software and Technology (UIST 96)*, pp. 95–96, 1996.

[12] S. Frees, G. D. Kessler, and E. Kay. PRISM interaction for enhancing control in immersive virtual environments. *ACM Transactions on Computer-Human Interaction*, 14:2–es, 2007.

[13] T. Grossman and R. Balakrishnan. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. *CHI '05 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 281–290, 2005.

[14] T. Grossman and R. Balakrishnan. The Design and Evaluation of Selection Techniques for 3D Volumetric Displays. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST 06*, pp. 3–12, 2006.

[15] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. A survey of design issues in spatial input. *UIST '94 Proceedings of the 7th annual ACM symposium on User interface software and technology*, pp. 213–222, 1994.

[16] R. Kopper, F. Bacim, and D. A. Bowman. Rapid and Accurate 3D Selection by Progressive Refinement. *Proceedings of the 2011 IEEE Symposium on 3D User Interfaces (3DUI 2011)*, pp. 67–74, 2011. doi: 10.1109/3DUI.2011.5759219

[17] R. Kopper, F. Bacim, and D. A. Bowman. Rapid and Accurate 3D Selection by Progressive Refinement. *Proceedings of the 2011 IEEE Symposium on 3D User Interfaces (3DUI 11)*, pp. 67–74, 2011.

[18] R. Kopper, D. A. Bowman, M. G. Silva, and R. P. McMahan. A Human Motor Behavior Model for Distal Pointing Tasks. *International Journal of Human-Computer Studies*, pp. 603–615, 2010. doi: 10.1016/j.ijhcs.2010.05.001

[19] J. J. LaViola, E. Kruijff, R. P. McMahan, D. Bowman, and I. P. Poupyrev. *3D User Interfaces theory and practice*. Pearson Education (US), New Jersey, United States, 2nd ed., 2017.

[20] J. Liang and M. Green. JDCAD: A Highly Interactive 3D Modeling System. *Computers and Graphics 18(4)*, p. 499506, 1994.

[21] K. C. Lode Vanacken, Tovi Grossman. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. *Proceedings of the 2007 IEEE Symposium on 3D User Interfaces (3DUI 07)*, pp. 117–124, 2007. doi: 10.1145/1166253.1166257

[22] J. F. Lucas. Design and evaluation of 3d multiple object selection techniques. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 2005.

[23] D. P. Mapes and J. M. Moshell. A Two-Handed Interface for Object Manipulation in Virtual Environments. *Presence: Teleoperators and Virtual Environments*, pp. 403–416, 1995. doi: 10.1162/pres.1995.4.4.403

[24] M. R. Mine. Virtual Environment Interaction Techniques. pp. 119–122, 1995.

[25] M. R. Mine. ISAAC: A Meta-CAD System for Virtual Environments. *Computer-Aided Design*, pp. 547–553, 1997. doi: 10.1016/S0010-4485(96)00095-4

[26] A. Olwal and S. K. Feiner. The Flexible Pointer: An Interaction Technique for Augmented and Virtual Reality. 2012.

[27] M. Ortega. Hook: Heuristics for selecting 3D moving objects in dense target environments. *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings*, pp. 119–122, 2013.

[28] J. S. Pierce, A. S. Forsberg, M. J. Conway, S. Hong, R. C. Zeleznik, and M. R. Mine. Image Plane Interaction Techniques in 3D Immersive Environments. *Proceedings of the 1997 ACM Symposium on Interactive 3D Graphics (I3D 97)*, pp. 39–44, 1997.

[29] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in VR. *Proceedings of the 9th annual ACM symposium on User interface software and technology - UIST '96*, pp. 79–80, 1996.

[30] R. Raskar, G. Welch, and H. Fuchs. *In First IEEE Workshop on Augmented Reality (IWAR98*.

[31] K. Riege, T. Holtkamper, G. Wesche, and B. Frohlich. *Proceedings of the IEEE conference on Virtual Reality (VR '06)*, p. 124.

[32] U. Schultheis, J. Jerald, F. Toledo, A. Yoganandan, and P. Mlyniec. Comparison of a Twohanded Interface to a Wand Interface and a Mouse Interface for Fundamental 3D Tasks. *IEEE Symposium on 3D User Interfaces (3DUI 2012*, pp. 117–124, 2012. doi: 10.1109/3DUI.2012.6184195

[33] R. Stoakley, M. J. Conway, and R. Pausch. Virtual Reality on a WIM: Interactive Worlds in Miniature. *Proceedings of the 1995 ACM Conference on Human Factors in Computing Systems (CHI 95)*, pp. 265–272, 1995. doi: 10.1145/223904.223938

[34] L. Vanacken, T. Grossman, and K. Coninx. Exploring the Effects of Environment Density and Target Visibility on Object Selection in 3D Virtual Environments. *2007 IEEE Symposium on 3D User Interfaces*, 2007. doi: 10.1109/3DUI.2007.340783

[35] H. P. Wyss, R. Blach, and M. Bues. iSith-Intersection-Based Spatial Interaction for Two Hands. Proceedings of the 2006 IEEE Symposium on 3D User Interfaces. *Proceedings of the 2006 IEEE Symposium on 3D User Interfaces (3DUI 06)*, pp. 59–61, 2006. doi: 10.1109/VR.2006.93