TOWARD GENERAL PURPOSE 3D USER INTERFACES: EXTENDING WINDOWING SYSTEMS TO THREE DIMENSIONS

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Forrest Reiling

June 2014

COMMITTEE MEMBERSHIP

TITLE:	Toward General Purpose 3D User Interfaces: Extending Windowing Systems to Three Dimensions
AUTHOR:	Forrest Reiling
DATE SUBMITTED:	June 2014
COMMITTEE CHAIR:	Professor Zoë Wood, Ph.D., Department of Computer Science
COMMITTEE MEMBER:	Professor Chris Lupo, Ph.D., Department of Computer Science
COMMITTEE MEMBER:	Professor Franz Kurfess, Ph.D., Department of Computer Science

ABSTRACT

Toward General Purpose 3D User Interfaces: Extending Windowing Systems to Three Dimensions

Forrest Reiling

Recent growth in the commercial availability of consumer grade 3D user interface devices like the Microsoft Kinect and the Oculus Rift, coupled with the broad availability of high performance 3D graphics hardware, has put high quality 3D user interfaces firmly within the reach of consumer markets for the first time ever. However, these devices require custom integration with every application which wishes to use them, seriously limiting application support, and there is no established mechanism for multiple applications to use the same 3D interface hardware simultaneously. This thesis proposes that these problems can be solved in the same way that the same problems were solved for 2D interfaces: by abstracting the input hardware behind input primitives provided by the windowing system and compositing the output of applications within the windowing system before displaying it. To demonstrate this it presents a novel Wayland compositor which allows clients to create 3D interface contexts within a 3D interface space in the same way that traditional windowing systems allow applications to create 2D interface contexts (windows) within a 2D interface space (the desktop), as well as allowing unmodified 2D Wayland clients to window into the same 3D interface space and receive standard 2D input events. This implementation demonstrates the ability of consumer 3D interface hardware to support a 3D windowing system, the ability of this 3D windowing system to support applications with compelling 3D interfaces, the ability of this style of windowing system to be built on top of existing hardware accelerated graphics and windowing infrastructure, and its

ability to support unmodified 2D interface applications windowing into the same 3D windowing space as the 3D interface applications. This means that application developers could create compelling 3D interfaces with no knowledge of the hardware that supports them, that new hardware could be introduced without needing to integrate it with individual applications, and that users could mix whatever 2D and 3D applications they wish in an immersive 3D interface space regardless of the details of the underlying hardware.

ACKNOWLEDGMENTS

Thanks to:

- My advisor Zoë Wood, for all of her guidance and wit.
- The wonderful people in the Wayland and QtWayland communities, without whom I would not have a functioning prototype.
- My parents, my family, and my girlfriend Katy, for supporting me always.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

The space we exist in is three dimensional, and this pervades every aspect of our interaction with reality. Everything we touch, see, and hear behaves according to the rules of this 3D space and this has made humans exceptionally proficient at reasoning about it, navigating through it, and modeling the behavior of things within it. Yet when we interact with our computers, an increasingly important part of our everyday lives, most of us do so exclusively through two dimensional user interfaces.

1.1 Two Dimensional User Interfaces

The two dimensional space in which we interact with computers has come to define these interactions in the same way that the three dimensional space in which we exist defines our interaction with reality. We use our fingers or a mouse press 2D buttons and open 2D menus, driving change in the applications 2D interfaces which the windowing system composites into a 2D image to be sent to a 2D display. While this is natural for some intrinsically 2D concepts, like documents and images, other concepts which have no intrinsic spatial embedding, like file systems and networks, are also embedded in 2D when they are presented to the user in order to allow users to reason about them spatially. Even in applications which are intrinsically 3D, like physical simulation and modelling tools, the 3D

application space is disjoint from the space in which the user exists (by necessity, since the application has no knowledge of the 3D relationship between the display and the user) and interaction between the 3D application space and the 3D user is traditionally done with 2D input events and 2D images.

The flat nature of contemporary graphical user interfaces has come to define not just the way we interact with applications, but has also become an important factor in the physical design of the computers that run these applications. This is particularly apparent in the mobile computer space, where cost, weight, display power, and mobility concerns push devices toward ever smaller physical profiles, while usability concerns drive the devices toward larger interface surfaces, leading the profile of mobile computers to become flattened against their displays, with the devices acting as a physical embedding of the 2D computer interface within the 3D space in which the computer exists. This forces users to make a tradeoff between the physical profile of their device and the usable size of their interface; larger displays drive up mass both directly and through the need for a larger battery to meet increased power demands, but a smaller displays limit the usable size of the human-computer interface which limits the usability of the device [citation needed]. In desktop computers the same tradeoff must be made, because even absent power and weight concerns the size of the interface is still constrained by the cost of the displays and the physical space needed to mount them in view of the user.

Two dimensional user interfaces are certainly not all bad. There is a natural analog between interacting with documents, images, and folders on a desk and interacting with their digital counterparts on a 2D display (which forms the underpinnings of the familiar desktop metaphor). 2D interfaces also map well onto commercially available display hardware as a result of the two co-evolving for sev-

eral decades, which keeps the hardware cost of 2D interfaces relatively low. 2D interfaces are mature and well studied, and there is a rich software ecosystem surrounding them which includes sophisticated, full featured user interface toolkits and advanced windowing systems, as well as a broad set of end user applications that provide 2D graphical frontends for almost every task a user need perform on a computer. Users are also familiar with the operation of 2D interfaces, which greatly reduces the time needed for users to learn new applications and improves their productivity with existing applications. There are certain applications, like document and photo editing and command line interaction, which fit well with 2D interfaces, and in these applications moving away from 2D interactions would likely be detrimental [citation needed?]. However, many applications are intrinsically 3D, or are not intrinsically spatial at all and are embedded in a 2D space because it is simple and well supported, and a transition to 3D interaction has the potential to greatly improve the usability of such applications [citation].

1.2 Three Dimensional User Interfaces

The hardware, software, and theory surrounding high quality 3D human-computer interaction has been the subject of academic research for many decades, and the improved spatial reasoning this provides has been demonstrated to improve usability in a number of applications [citation]. 3D user interfaces are a broad group of interaction paradigms, including everything from desktop 3D modeling with a mouse and keyboard to fully immersive virtual reality. This thesis focuses on immersive 3D interfaces, which is used here to refer to 3D interfaces in which the user perceives the 3D interface elements to be in the same 3D space as their body, and has some way of manipulating these interface elements in 3D with corresponding 3D motion by some part of their body. The hardware needed to

support these types of interfaces has traditionally been very expensive, but recent technological improvements in a number of fields have brought many of the core hardware technologies onto the consumer market, bringing both high quality 3D input devices and immersive 3D displays into the reach of everyday computer users.

1.2.1 3D Input Devices

Early 3D input devices to come into the consumer 3D interaction market were largely marketed as video game accessories, though their availability has led to their use in a wide variety of other application. These devices can be broadly categorized into two groups: devices which resolve the position and/or orientation of an element held or worn by the user, and range-imaging cameras, which produce a 3D image of a passive scene which contains no sensing elements itself.

The first category had its first commercial success in consumer markets in 2006, when Nintendo introduced the Wii Remote, or Wiimote, as the primary controller for its new Wii entertainment system. This controller, unlike traditional game console controllers, was able to sense its position, orientation, and acceleration along 3 axes. The Wiimote provided limited 3D control within Wii games, and was soon adopted by developers for a wider variety of tasks, including controlling the visualization of volumetric medical data [citation] and enabling head tracking 3D on traditional displays [citation]. Several devices which provided similar input using a variety of different tracking technologies soon came to market, including Sonys Playstation Move in 2009, and the Razer Hydra in 2011 (Produced by Sixense Entertainment in partnership with Razer USA). Until the time of this writing all commercially available, consumer grade, 3D tracking solutions have been handheld controllers gripped by the user, but two new multi-

sensor, wearable, full-body tracking solutions (Sixenses new magnetic tracking system, STEM, and PrioVRs inertial tracking system) are due to become commercially available within the next year in response to an emerging consumer virtual reality market.

Range imaging cameras can be based on a variety of technologies, many of which have been commercially available for many years but have been too expensive to be accessible to a wide body of consumers. In 2009, following advances in real-time structured-light 3D scanning by Primesense Ltd, Microsoft released a range imaging camera based on a Primesense sensor to the public, under the name Kinect, as an accessory for their Xbox 360 game console. Designed to perform full body tracking in 3D on multiple users, the Kinect enabled a variety of new interaction techniques in Xbox games. Like the Wiimote, the Kinect was quickly adopted by third party developers and applied to numerous non-gaming domains, including robotics applications like Simultaneous Location and Mapping [citation], and a variety of medical applications [citation]. Although the Kinect has received a great deal of attention, being the first consumer grade sensor capable of producing high quality images in real time, many other sensors have since come to market which offer comparable or better performance in a variety of applications and operating conditions. Primesense Ltd., the company that developed the technology underpinning the first generation Kinect, also developed sensors based on the same technology that were released both directly by Primesense under the name Carmine, and through Asus as the Xtion and Xtion Live, which all offer very similar performance to the first generation Kinect [citation]. Very recently, several consumer-grade range imaging cameras have become commercially available which rely on time-of-flight technology, which has several advantages over structured lighting including lower software overhead, faster response time, and

better bright light performance [citation]. This includes Microsofts next generation Kinect, released with the Xbox One in 2013, and the DS310 and DS325 from Belgium based SoftKinetic. The Softkinetic DS325, also sold rebranded as the Creative Senz3D, is designed for close interaction and finger tracking rather than full body tracking [citation], and competes in the consumer market with the Leap Motion, a desktop stereo camera designed specifically for hand, finger and stylus tracking in the space immediately above the keyboard. Several other companies, notably Texas Instruments and pmdVision, provide Time of Flight solutions, but to the authors knowledge they do not sell consumer time of flight products as of the time of this writing. This is by no means an exhaustive list of 3D input devices; it is meant only to demonstrate the growing diversity of 3D input devices reaching the consumer market, and that this is a relatively recent development. At the surface level, these devices appear to produce a wide variety of input, but when constrained to human-computer interaction applications it becomes apparent that simple input models can capture the useful input produced by all of these devices. Essentially this is because the only mechanism humans have to produce 3D input is the movement of their body through the 3D space or the use of this movement to move an object through 3D space, which can be captured, respectively, by the notions of skeletal tracking and 3D pointing devices [cite Jester].

1.2.2 Immersive 3D Displays

The term 3D display has come to refer to a very broad category of devices, so the term immersive 3D display is used here to refer to graphical displays which support both stereo parallax (rendering the scene from a separate viewpoint for each eye) and head-tracking motion parallax (adjusting the position and orientation of

these viewpoints based on the 3D relationship between the user and the display), as both are required to create a convincing 3D experience for the user (This is discussed in more detail in the background section). This excludes commercial 3D televisions and 3D movie theaters because they do not provide head tracking, and excludes haptic and audio displays because they are not graphical. There have been many systems which meet these requirements in research and industrial applications, including Responsive Workbenches [citation], Hemispherical Displays [citation], CAVE Automated Virtual Environments (CAVEs) [citation], and Head Mounted Displays (HMDs) [citation], and some of these technologies, particularly CAVEs and HMDs, have received significant research and developments, allowing the technology to mature significantly. Most of these technologies have remained outside of consumer reach, largely due to the large size and high cost of such systems, with the exception of HMDs, whose simplicity and compact design has led them to enjoy intermittent commercial availability for many years. A comprehensive discussion of commercially available HMDs is outside the scope of this paper, but it is worth noting that the recent release of OculusVRs Oculus Rift development kit to consumer markets has sparked a resurgence in interest in virtual reality for video games and other entertainment applications, leading to the announcement of several new consumer HMDs, including a consumer version of the Oculus Rift [citation], Sonys Project Morpheus [citation], and True Player Gears Totem [citation].

Priced at only a few hundred dollars, these HMDs put high resolution, wide field-of-view, immersive 3D display technology in the hands of everyday computer users, and the continuing advances of consumer graphics processing hardware gives them the ability to drive convincing 3D scenes onto these displays with commodity hardware. Furthermore, like 3D input devices, the similarity in

function between these devices means their behavior can be captured abstractly by relatively simple input and output models.

CHAPTER 2

Motivation

The recent influx of commercially available 3D input devices and immersive 3D displays to the consumer market, coupled with high performance consumer graphics hardware, has given end users access to all of the hardware needed to support high quality, immersive 3D human-computer interaction for the first time ever. However, current application support of this kind of hardware is very limited, consisting mainly of demo applications and video games, and applications which support more than one of these devices are even more rare.

2.1 Obstacles Facing the Adoption of 3D Interfaces

In general, immersive 3D user interfaces require both a good 3D input device and an immersive 3D display, and with such limited device support it is rare that an application supports both and even more rare that an end user will own a 3D input device and an immersive 3D display which are both supported by the 3D user interface application they wish to use (much less every 3D user interface application they wish to use). This problem could be attributed to many factors, including that it is too early in the life of these devices for applications to have been developed, or that there is simply limited application potential for this these devices. And while these certainly could be contributing factors, there are also tangible shortcomings in the software ecosystem surrounding this new hardware

which indicate that this problem is not simply going to go away on its own.

2.1.1 Device Abstraction

The first problem to become immediately apparent is the sheer diversity of 3D interface devices, a problem which will only get worse as more and more devices come to market. There is a fair amount of similarity within these devices, and the even greater similarity within the actual 3D interface primitives which each is actually capable of providing. Every 3D input device discussed above provides either some information about the 3D pose of the users body [cite Jester] or the 3D transform of some kind of handheld device, and immersive 3D displays all serve the same purpose of giving the user a sense of presence in a virtual (or mixed reality) 3D space. Despite this, there is no widely adopted abstraction for either 3D input devices or immersive 3D displays, and while some such abstractions exist, each has its own unique shortcomings (this is discussed further in the related works section).

Rather, every one of these devices comes with its own API, designed to work with that device and usually other devices from the same manufacturer. If an application wishes to use this device it must be ported to use that devices API, and if it wishes to be compatible with multiple devices in the same class from different manufacturers it must include dedicated code for each of the devices it wishes to support, and this code must be maintained by the developers of each application independently. Support for devices can be abstracted by a user interface toolkit like Vrui [citation], a video game engine like Unity or Unreal (via vendor provide plugins), or even a dedicated abstraction libraries like MiddleVR [citation] or VRPN [citation]. Each of these has its own strength and weaknesses, but there are also overarching shortcomings of including the abstraction layer

in toolkits used on a per application basis. First, this means that the device abstraction layer has to be replicated for each toolkit (causing the same problems as replicating it for each application). This could hypothetically be resolved by the uniform adoption of a single toolkit which meets the need of every application needing a 3D user interface, but given the wide variance in demands between something like a 3D file browser and an immersive VR experience, this seems both unrealistic and, in the authors opinion, very much undesirable. Secondly, if the abstraction is done within toolkits used on a per application basis, then two applications using different toolkits (or perhaps even the same toolkit) that attempt to use the same device simultaneously could block one another from accessing the device. This is closely related to the next major problem with the software ecosystem surround 3D user interface devices.

2.1.2 Multiple Application Support

The ability to use multiple applications simultaneously has become a core feature of the interface paradigms we use today, and the ability of a user to install and run together whichever set of applications they like is the key point of software modularity that has allowed personal computers to be useful to a broad class of users with highly diverse requirements.

This same point of modularity can be applied to 3D user interfaces, and to a certain extent it already is. It is certainly possible to install multiple 3D user interface applications and, depending on the applications, maybe even run them simultaneously. However, there are serious limitations here as well, particularly when it comes to immersive 3D displays. These displays require custom projection of a 3D scene for each eye, and this projection must be consistent with the 3D position of the users head relative to this scene and the display surface,

and many HMDs require a post-projection adjustment to correct for distortion introduced by the optical system (this is discussed in detail in the background section). While it is relatively straightforward to implement this behavior in an application which draws itself (and only itself) to the entire display surface, sharing the 3D display between multiple applications with 3D user interfaces introduces significant problems. The essential problem is that in order for the 3D interface space to be divided between multiple 3D interfaces from different applications in a meaningful way, it must be divided in 3D. This is difficult because current graphics and windowing infrastructure, as well as the 2D display technology underlying the immersive 3D display, is designed around the paradigm of applications producing 2D output which is combined in 2D by the windowing system and driven onto the 2D interface space of a traditional display. This works well for dividing the 2D space of a display among multiple 2D interfaces, since they can each be given a rectangular region of the rectangular display, but dividing the 2D display surface of an immersive 3D display among multiple 3D interfaces in the same way (without applying the correct stereo projection and optical distortion correction) produces results which do not appear to be in the same 3D space.

This means that while an immersive 3D display can produce a compelling 3D interface space for a single application, it is not possible for multiple 3D interface applications to share the 3D display in the same way that 2D applications can share a 2D display. It also means that 2D applications which have no reason to need a 3D interface are also unable to use the immersive display, despite the fact that embedding a 2D interface surface in a 3D interface space is conceptually simple and well defined.

2.2 Insights from 2D User Interfaces

The core goal of this thesis is derived from the simple observation that the problems currently facing the development of applications with 3D user interfaces and the integration of the hardware that supports them are present for 2D interfaces as well, with the key difference that in the domain of 2D interfaces these problems have already been solved. Despite the fact that the diversity of displays, mice, and keyboards dwarfs the diversity of 3D user interface devices, users are able to assemble a hardware interface from almost any combination of devices they like and run all of their favorite applications on top of their custom hardware interface. New 2D interface devices need not be integrated into every application that uses them, and multiple 2D interfaces from different applications can be used together in the same 2D interface space in arbitrary combinations.

2.2.1 Windowing Systems

Applications with 2D interfaces no longer suffer these problems is because modern consumer operating systems provide a set of 2D interface abstractions called a windowing system. Windowing applications do not interact directly with the mouse, keyboard, or display. Rather the windowing system manages input devices and displays (usually through lower level abstractions provided by the kernel), and provides the interface capabilities of these devices as services to applications. Applications receive input events like mouse movement from the windowing system abstractly without needing any knowledge of what type of mouse is used, how it is connected, or who manufactured it. The 2D images produced by applications are not drawn directly to the display, they are given to the windowing system which then composites the output of all running applications (sometimes in a sep-

arate compositor program, depending on the windowing system) into a final 2D image which is scanned out to the display itself. This basic system architecture is present, with slight variation, in every major graphical operating system. It is connected with the prevalent Windows, Icons, Menus, Pointer (WIMP) interaction paradigm and the popular desktop metaphor, which are well understood, well tested, and familiar to users. This architecture has also strongly influenced the way applications interact with hardware accelerated 3D graphics systems, leading to a design pattern where applications are responsible for projecting their 3D content into a 2D image before delivering it to the windowing systems, and this has in turn influenced both the design of 3D graphics APIs as well as the hardware which underlies them. The ubiquity of windowing systems has also profoundly affected the high level topology of the software ecosystem surrounding WIMP interaction, leading to the emergence of user interface toolkits like QT and Java Swing that abstract popular windowing systems behind their common functionality so that sophisticated, cross platform, WIMP applications can be developed without knowledge of the underlying software mechanisms, much less the hardware devices, that support them.

Even existing 3D interface applications use the windowing system to abstract traditional input and to draw to the 2D display that underlies its immersive 3D display, but without the ability to abstract 3D input devices and to mix 3D interfaces in 3D, these windowing systems do not give applications the ability to share 3D interface hardware in a meaningful way.

2.3 Proposed Solution: A 3D Windowing System

The primary goal of this thesis is to demonstrate that windowing systems are capable of solving the some of the core problems facing 3D user interfaces in the same way that they have already solved the exact same problems for 2D user interfaces, and that this can be done with extensions to an existing windowing system, allowing both unmodified 2D applications and as device-agnostic 3D applications to window into the same 3D interface space.

The type of windowing system described here extends the concept of a window as a 2D region of a 2D interface space to the 3D interface space provided by the 3D user interface hardware described above. It allows 3D applications to create a 3D analog of a traditional window, representing a 3D region of the 3D interface space which can be manipulated in 3D in much the same way as a traditional 2D window can be manipulated in 2D. These 3D windows can listen for 3D input events via the same mechanism that is used to listen to 2D input events, and the 3D output they produce is mixed in 3D with the 3D output of other 3D applications.

Additionally, this type of windowing system allows traditional, unmodified 2D applications to create a 2D interface context in this 3D windowing space which behave exactly the same as a normal window from the applications perspective. The windowing system embeds these 2D windows in the space in much the same way that a sheet of paper embeds a 2D document in 3D reality, allowing the user to manipulate and manage these windows as 3D objects. 3D input events managed by the windowing system are projected onto the 2D window before being delivered to the 2D application, allowing the user to send meaningful 2D input to the application with a 3D input device.

2.3.1 Advantages of This Approach

There are numerous advantages to solving these problems for 3D user interfaces in the same way that we solve them for 2D interfaces, a few of which are discussed here in detail. Some of these advantages are the same advantages that led to the adoption of windowing systems in the first place, and others are simply result from leveraging extensive work put into windowing systems for 2D interfaces. This is by no means meant to be an exhaustive list.

Hardware Abstraction and Multiple Application Support

This approach allows a hardware 3D interface (consisting of at least one immersive 3D display and at least one 3D input device) to support a unified 3D interface space, where both 2D and 3D applications are treated as first class components of the human-computer interface and managed together in the 3D space via a unified window management mechanism.

This means that any hardware capable of supporting the 3D windowing system can support all 3D applications which use it (as is the case with 2D applications), and that new hardware need only provide a driver for the windowing system abstraction to achieve support from all applications using the system (as is also the case with 2D interfaces).

It also means that the structure of the software ecosystem surrounding 2D WIMP interaction can be applied to the software ecosystem surrounding 3D interfaces, allowing the development of a wide variety of user interface toolkits which provide high-level, domain-specific interaction metaphors built on top of common abstractions provided by the windowing system, allowing multiple applications using different toolkits (or no toolkit at all) to share the 3D interface

hardware supporting the system in a meaningful way. Furthermore, because the system supports unmodified 2D applications, support for 3D interface elements could even be integrated into existing 2D interface toolkits where appropriate.

Compatibility With Existing Graphics and Windowing Infrastructure

As the provided implementation demonstrates, it is possible to support compositing 3D content in a 3D space while only needing to send 2D images from the application to the windowing system. This means that existing, full-featured 3D graphics APIs, which give the application full control over every aspect of how its 3D content is drawn into a 2D image, are still perfectly well suited to this task. This means that applications retain full flexibility in what they draw and how they draw it, and can still benefit from acceleration by high performance consumer graphics processing units (GPUs). It also means that 3D applications still benefit from the extensive software infrastructure that has been put in place to allow 2D applications to efficiently pass 2D images to the windowing system and to allow the windowing system to composite these images off screen. Together this means that the 3D windowing system can efficiently support both 2D and 3D applications without needing to lay extensive new infrastructure to do so.