Final Project Report

Jonathan Sumner Evans*, Robinson Merillat†, and Sam Sartor‡
Department of Computer Science, Colorado School of Mines
Golden, Colorado
Email: *jonathanevans@mines.edu, †rdmerillat@mines.edu, ‡ssartor@mines.edu

***Abstract*—Virtual Reality (VR) Technology has been developing rapidly over the past decade. Current solutions such as Unity attempt to use old programming languages and even older paradigms to implement VR applications and thus, limit developers' abilities to create new and unique environments. With every cutting edge technology, new paradigms and design patterns must be invented. In this paper, we discuss a deferred immediate mode (DIM) application architecture suitable for implementing large virtual reality applications. We present a UI library which utilizes this architecture and a few case studies of this library in use.**

I. Introduction

The Virtual Reality (VR) market is growing rapidly. The International Data Corporation (IDC) projects that revenues for the combined Augmented Reality (AR) and Virtual Reality markets will grow from $5.2 billion in 2016 to more than $162 billion in 2020 [4]. This flourishing new industry has created an exciting new field of Software Engineering with great potential for revolutionary new design paradigms and program architectures.

Most current frameworks and libraries attempt to apply old design paradigms and program architectures to virtual reality. While these paradigms and architectures are well-suited for 2D user interfaces and rendering 3D environments to a flat screen, they are not designed with VR in mind. Although endeavors to adapt these old patterns to VR have been successful, they are restrictive in both the code architecture and way of thinking. Grace Hopper loved to say "The most damaging phrase in the language is: 'It's always been done that way.'"[3] This quote promotes new ways of thinking. Utilizing outdated architectures hinders the exploration and the potential for new design paradigms and program architectures.

Our goal is to find a system that provides a modern, fast, and practical approach to virtual reality development. Specifically, we require that this system meets the following criteria.

A. Performant

Traditionally, animation is done around 60 frames per second. This is due to the wave of monitors and televisions that ran at 60Hz and had a refresh rate of 60 frames per second (fps). This is fine when looking at a screen from a distance, however it has been found that in order to appear believable and prevent disorientation, a virtual reality program must run with at least 90 fps. In fact many studies have shown that when VR is run with lower frame rates users experience headaches and nausea faster than when at high frame rates [5]. As VR headsets run two mini monitors concurrently, the effective required frame rate is 180fps. Achieving this frame rate is resource intensive and requires highly efficient and optimized code. Additionally, multi-threading is imperative so that long-running processes can occur without blocking the user interface. This is unlike a traditional desktop user interface where blocking the UI process for a second does not have a major affect on the usability of the program. These factors require a VR framework that is highly performant and multi-threadable.

B. Natural

A *natural user interface* is an interface that can be used without the need for a controller [7]. Although current VR systems utilize hand-held controllers, they emulate this goal much better than desktop, mobile, and web applications. For our discussion of these natural user interfaces, we define the following terms.

Definition 1. *A* planar UI *is a user interface where components are organized along a 2D surface.*

Definition 2. *A* spacial UI *is a user interface where components are organized within a 3D space.*

Both planar and spacial user interfaces are effective in virtual reality applications. This cannot be said of applications that use traditional human-interface devices such as mice and keyboards. Because virtual reality environments are inherently 3D, they make spacial UI convenient and practical for the first time. Thus, the ability to create spacial UI rather than merely to planar UI is a high priority.

C. Flexible

Computer software is used to solve a variety of unique problems which in turn require a variety of user interface solutions. Thus, a good user interface toolkit must be general enough to accommodate the goals of the developers writing the software. Most importantly, such a toolkit must not be opinionated about which types of application should be created with it. The desktop, mobile, and web application fields already have general purpose user interface toolkits (e.g. HTML and GTK). We need a UI library for VR which is equally general-purpose.

D. Modular

A direct result of flexibility is modularity—a flexible architecture cannot be a monolith. Many current VR libraries include features such as pathfinding and character rigging which must be included even if they are not used. This is not modular and most non-entertainment software does not require any of these features.

From a software engineering standpoint, a good UI library must allow the programmer to integrate any number of components, but these components should be add-ons, not dependencies. Additionally, these components should be compartmentalized and not interfere with one another. This library design promotes good software engineering practices including the UNIX philosophy ("do one thing and do it well") and the open/closed principle ("software entities should be open for extension but closed to modification").

We need a framework that has a minimal feature set baked in while still allowing extensibility via the addition of self-contained modules.

***

Given these criteria, we evaluate current program architectures and UI libraries including Unity, entity component systems, React-like architectures, and event tree architectures in Section II. We also describe how these architectures influenced our deferred immediate mode architecture which we formally present in Section III. In Section IV we describe Flight—an implementation of a VR UI library using the deferred immediate mode architecture. Then we present three case studies of Flight and the deferred immediate mode architecture in use in real applications in Section V and conclude in Section VI.

II. Evolution of Deferred Immediate Mode

We researched many current VR software architectures to find one which suited our needs. All of them had many advantages but also many disadvantages. In this section we describe a variety of libraries and evaluate their ability to accomplish our goals as described in the previous section.

A. Unity

Unity is a game engine which was designed to allow programmers to easily create games and has many features which make this process effortless. Unity has been used to create many successful, award-winning VR games and applications including SUPERHOT [1] and TiltBrush [6]. The Unity ecosystem is growing rapidly and has become the de facto standard for building VR applications. For example, Google released a TiltBrush toolkit on GitHub under the Apache 2.0 library [2].

However, Unity's UI system was designed for building 2D UIs and has been adapted for making 3D UIs. Although these adaptations have been successful, we wanted to explore systems which have 3D UI elements as first class citizens.

Flight is our implementation of a VR UI library using the DIM architecture. It is written in Rust and is designed to be highly modular. It uses Rust `FnOnce` closures to implement the deferred aspect of the DIM architecture and resolves state using a *guru* system.

A. Language

We chose to implement Flight in Rust for a few reasons.

1) Performance: As mentioned in Section I-A, virtual reality requires a high frame rate.

2) Type System:

B. Dependencies

The Rust dependency manager, Cargo, allows programmers to easily include third-party libraries from https://crates.io. To avoid duplicating work by other programmers, we utilized a few external libraries for Flight. The main dependencies are listed below.

- **`WebVR`**: an OpenVR wrapper built by Mozilla for the Servo browser.
- **`nalgebra`**: library for geometric/volumetric operations and queries.
- **`gfx`**:

[1] Joe Durbin. *Superhot VR Wins Best Game At Unity Vision Summit 2017*. May 1, 2017. URL: https://uploadvr.com/superhot-vr-best-game/.

[2] googlevr. *Tilt Brush Toolkit*. URL: https://github.com/googlevr/tilt-brush-toolkit.

[3] Grace Murray Hopper. *Information Week Interview*. Mar. 9, 1987. URL: https://quoteinvestigator.com/2014/11/27/always-done/.

[4] IDC. *Worldwide Revenues for Augmented and Virtual Reality Forecast to Reach $162 Billion in 2020, According to IDC*. Aug. 15, 2016. URL: https://www.idc.com/getdoc.jsp?containerId=prUS41676216.

[5] *The Importance of Frame Rates*. URL: https://help.irisvr.com/hc/en-us/articles/215884547-The-Importance-of-Frame-Rates.

[6] Unity. *2015 Unity Awards Winners*. URL: https://unityweb.unity3d.com/awards/2015/winners.

[7] Alexandre Wimmers et al. *Natural User Interface and Virtual Reality Integration in Video Games*. Mar. 31, 2015. URL: http://csiflabs.cs.ucdavis.edu/~cs193/cs193_2015/23_design.pdf.