Final Project Proposal

Sumner Evans, Robbie Merillat, Sam Sartor

October 11, 2017

1 Introduction

VR Technology has been developing rapidly in the 21st century. Current solutions such as Unity attempt to use old programming languages and paradigms to implement VR environments and thus limit developers' abilities to create new and unique environments. With every cutting edge technology, new paradigms and design patterns must be invented. This project intends to explore and implement these paradigms and design patterns.

2 Objectives

The goal of this project is to experiment with a variety of paradigms and design patterns to see which work best for implementing VR environments. To do this, we will implement four different VR environments where users can move around and interact with a variety of virtual objects. Each team member will build one environment individually and the fourth will be built by the whole team and will connect the individually-built environments.

2.1 Environments and Interactions

The following sections describe the concepts for our four environments and the interactions that can occur in each environment.

2.1.1 Sam — Workbench

The goal of this environment is to find a general pattern for VR element design that can apply to both diegetic (semi-realistic) and spacial (purely abstract) items. Specifically, it includes logic for moving, placing, snapping, and recombining arbitrary VR elements. These elements are split into 2 categories. Some are purely abstract user interface items (spacial), others are blocks that appear more concrete (diegetic). The user has the ability to recombine blocks into moving objects such as miniature vehicles. More interestingly, the user can recombine interface elements into static control "devices" for these objects. For example, the user could attach 4 rocket engines together, and control it by combining the 4 thrust controls into a crude joystick.

2.1.2 Sumner — Snowflakes

Snowflakes will be a winter-themed environment where users can create their own structures by manipulating snow blocks. The environment will have "snow" on the ground and winter themed items in the environment such as snowmen. The user will be able to create new snow blocks by pointing both controllers at the ground and pulling both of the triggers. This action will create a snow block being held between the users hands. When the user releases the triggers, the snow block will fall in a physically-accurate manner with collision detection.

2.1.3 Robbie — Let's Get Physical

This project provides a physically "real" environment. That is an environment that applies some of the fundamental laws of physics as we know them. Simple gravitational pull, acceleration, and collisions which in turn includes momentum. To achieve this, the Rust `nphysics3d` library is used to provide physics utility functions to the program. Various objects will be placed around the environment that can be picked up, thrown, dropped, and/or interacted with in other ways. One example of such interactions would be shooting a bow and arrow.

2.1.4 Team — VRsh

This environment will be a user-customizable environment where users can:

1. Navigate to other environments using a "program picker".

2. Persist the state in other environments using an intuitive UI specialized for VR.

3. Customize the environment state using design motifs that are specialized for VR.

3 Proposed Implementation

- Programs do not themselves store state. They receive state from VRsh and then modify it.

- Programs register their default state with VRsh?

- Creating a new copy of a default environment is done by grabbing the environment from the program picker.

- VRsh stores all environment state information and when the user "reopens" a state, VRsh passes that state to the application.

4 Plan of Action

This project will have four main stages that correlate with the due dates for the individual assignments and the final project milestones.

> 10/18 **Individual Assignments** — Demo the individual environments
>
> 11/03 **Milestone I** — Building the VRsh environment
>
> 11/17 **Milestone II** — Integrating the VRsh and individual environments
>
> 12/08 **Final Code Submission** — Final touches to the VRsh environment.

In addition to the above code submissions, we will also produce a final report along with the Final Code Submission describing the work that we did and the lessons we learned while implementing the project.

4.1 C-MAPP Event Readiness

Our goal is to have this project ready for the C-MAPP event in January. To do this, we must have a finished product and report by the end of this semester.

5 References and Dependencies

Because of the performance requirements and complexity of virtual reality software, we decided to use the Rust programming language for all the projects in this independent study.

Our individual and final projects will all be built using our own shared, open-source library called `flight`, which will be continually updated to reflect what we learn while developing these projects. As of writing, `flight` incorporates our own VR hardware interface, several custom rendering environments, mesh manipulation tools, and a few asset loading utilities.

Although most of the planned features of our shell can be implemented using the Rust standard libraries, there are several other tools that we will be depending on. The most notable are `gfx` (a GPU graphics wrapper), `nalgebra` (vector and matrix math), `ncollide` (geometric operations such as collisions and intersections), `nphysics` (physics engine), OpenVR (Valve's VR hardware SDK), image (raster image loader), and futures (asynchronous events). With the exception of OpenVR, all these libraries can be found on `crates.io`.