The Rings of Mars

Brian Wilcox & Connor Lamon
Stanford University
Stanford, CA
brianwilcox@stanford.edu, conlamon@stanford.edu

Abstract

The Rings of Mars is an interactive Virtual Reality (VR) game using Three.js and WebVR. The goal of the game is simply to fly through all of the rings on the map in the shortest amount of time possible. From a technical perspective, the game is lightweight in comparison to the processing that goes into developing a game in Unity or Unreal. Additionally, this current game is compatible with various VR headsets and mobile devices.

1. Introduction

VR games have been developed for years dating back to the 1980s with the goal of providing an interactive environment for users on various platforms [3]. Recent developments in compute power and graphics have lead to more VR innovation in recent and now it has been a hot-topic in the public eye. Many of these VR experiences are known to have a large overhead just to use them. This comes in the form of acquisition cost, power availability, and graphic capabilities. Therefore, there is a need for lightweight VR experience development that can be created on devices that have low technical capabilities which the common user can be interface with without extra cost. This has inspired the development of The Rings of Mars, an interactive first-person flight game that can on various devices (industry VR headsets and mobile phones). Since the group was using the View-Master for this project, they calibrated the IMU output to work well with Google Cardboard that has similar view parameters.

2. Methodology

The group implemented the game using Three.js and C++ code in the Arduino interface provided from EE267: Virtual Reality [2]. This was done mainly to interface with the IMU to maintain head-tracking during the game. On top of this, the group used libraries Three.VREffect and WebVR to manage the camera and provide an easy-to-integrated VR stereo visual. The display parameters used for this effect are from Google Cardboard 2015 but these display parameters can be configurable for several other popular headsets in the VR space. The group also generates a 3D texture map using the tutorial provided Los Angeles Times Data Desk [1]. This map uses a cubic mesh to generate a 3D rendering of a desert-like planet (such as Mars).

The controls of the game are mapped to have two simple components. The first component is to tell the camera whether or not it should move forward. To do this, one simply must press the 'w' key on their keyboard. The second component is camera pitch adjustment using the headset controller. This allows the first-person view to move vertically with the eyes of the user. In order to print progress on the game to the screen, jquery is used to interface with Three.js variables that keep track of the positions of the next ring and the total number of rings encountered/left to pass through.

2.1. Demo

Screenshots of the game are shown in Figure 1, 2, and 3.

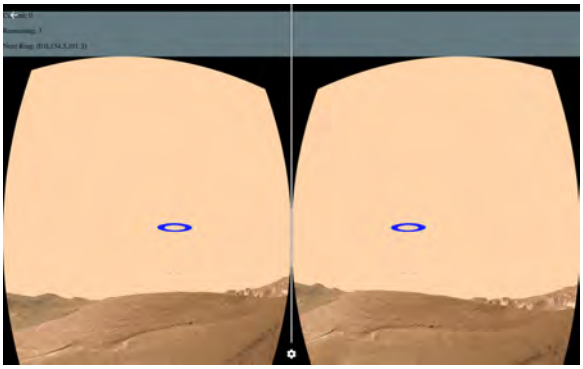

Figure 1. Stereo Mode

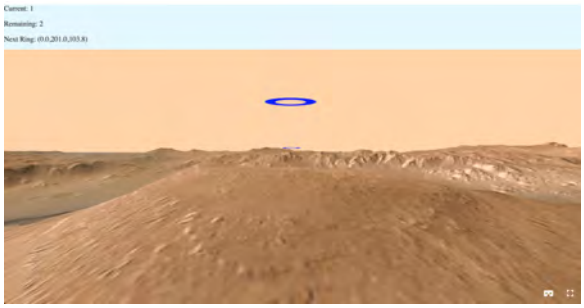

Figure 2. Game View

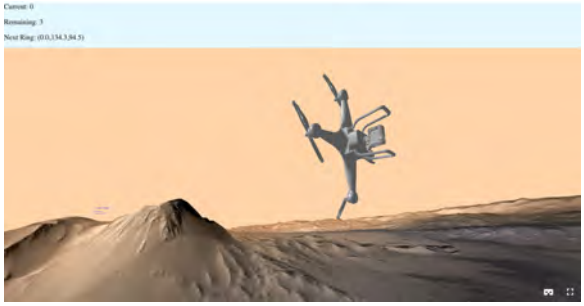

Figure 3. Drone View

3. Other Attempts and setbacks

There was one major setback that came with using three.js and WebVR. The main issue was that the rendering of the large map in stereo mode had latency issues on an older Macbook Pro as well as an older Dell laptop (3 and 4 years old respectively). Even trying to play a game in Unity lead to major problems on both laptops as the frame rate dropped constantly. Had the group had more time, they would have tried to find a way to upgrade their development and demo hardware.

Ideally, the team wanted to develop a third-person game such that movement would affect the position of a drone along with the camera. Additionally, the group would have liked to incorporate horizontal motion in the head-tracking scheme. Additionally, they would have liked to adjust the direction of translation of the drone in a four directional interface using the pitch and roll of a second controller. Due to time constraints, the latency issue, and issues interfacing older versions of WebVR and Three.js with newer JavaScript libraries, these features were not met. Some additional problems the group came across was differences in camera parameter specifications between the installed library for the display parameters in the headset configuration (Google Cardboard 2014 in WebVR) and the actual headset display parameters (Google Cardboard 2015 in the ViewMaster). This lead to some nausea with horizontal head-tracking.

With more time, the group would try to address many of these issues and also ensure that libraries used are up to date with the latest available ones. Additionally, more complexity would be added to the game levels for a more enticing VR experience.

References

[1] L. A. T. D. Desk. datadesk/vr-interactives-three-js, 2017.

[2] G. W. et al. Ee 267 homework 6: Pose tracking, 2018.

[3] M. Gotsis. Games, virtual reality, and the pursuit of happiness, Sept 2009.