Natural User Interface and Virtual Reality Integration in Video Games

Last Revised: January 31, 2015

Alexandre Wimmers adwimmers@ucdavis.edu
Mario Yepez myepez@ucdavis.edu
Timothy Tong tktong@ucdavis.edu
Valerie Gadjali vgadjali@ucdavis.edu

1 Summary

Since its birth, video games have continuously evolved — finding new gameplay, integrating new mediums, etc.

The traditional real-time strategy (RTS) game implements multiple dimensions using a layered style of interactions – typically land and sky. Units exists in one layer at a given time and majority of units can only interact with one layer. Only a few are capable of existing in one layer and interacting with both layers. This emulates a three dimensional field, but restricts the z-axis to only two domains. It is important to note that the multiple dimension emulation is on the game implementation and not the graphics render. This presents the initial question – is it possible to have a truly three dimensional real-time strategy game?

Throughout the years, real time strategy games have not changed much aside from the multiple layered system. Almost all real-time strategy games released can be seen to follow this standard model. The only differences being textures, models, sounds, names, and in-game stats. It is undoubtable that the real-time strategy game industry is struggling to provide the gaming community with a more enjoyable experience and greater immersion. Gaming news articles such as the following have begun emerging in recent years:

- Is the Real Time Strategy Genre dead?
- Face Off: Is the RTS genre dying?

Based on the emergence of virtual reality and natural user interface technologies though, this challenge may very well be addressable. The traditional keyboard and mouse with a monitor is incapable of interacting with three dimensional space easily. The only possible way is to keep one axis constant and then move along the other two. A natural user interface, however, receives input in three-dimensional space from the physical world. Virtual reality is then used to aid the player immersion and camera perspective. The question now becomes – is it possible to integrate virtual reality and natural user interface to create a truly three dimensional real-time strategy game?

2 Stakeholders

The stakeholders are Alexandre Wimmers, Mario Yepez, Timothy Tong, Valerie Gadjali, Aveek Das, Dr. Prasant Mohapatra, and Dr. Xin Liu.

Course instructors and assistants – Aveek Das, Dr. Prasant Mohapatra, and Dr. Xin Liu – shall act as advisors overseeing project progress and providing support.

Because this is a student proposed and self-funded[1] project under course 193AB in the Computer Science Department at the University of California, Davis, the decision making shall reside with the students of this project – Alexandre Wimmers, Mario Yepez, Timothy Tong, and Valerie Gadjali. A single signature from any member of the team will signify client approval of this document.

3 External Ecosystem

3.1 Hardware/Software Requirements

The hardware requirements are a minimum of two computers, two Leap Motions, and two Oculus Rifts. This project implements multiplayer capabilities, therefore a minimum of two of each devices is required.

- The Leap Motion is a device specialized in receiving hand gestures at 1/100 mm precision and at over 200 frames per second. In general, It was chosen over the Kinect 2 because of its specialization, cost effectiveness, and integration with both the Oculus Rift and Unity game engine.
- The Oculus Rift is the most well-known and popular virtual reality device providing users with a deep immersive experience. With the release of the Oculus Rift Development Kit 2, there is positional tracking and improved resolution. The Oculus Rift is also supported on the Unity game engine.

The software requirements are Unity, Blender, Oculus SDK, and Leap Motion SDK.

- Unity is a freemium[2] game engine using either UnityScript[3] or C#. For this project's purposes, the free version will suffice since it provides support for Leap Motion and Oculus Rift. It is available for development on OSX and Windows, but deployable on all major systems as well as browsers.
- Blender is a free open source 3D graphics software. It will be used to create visual effects and models. Blender is available on all major systems and resulting models are exportable to Unity game engine.
- Oculus SDK is the source development kit for the Oculus Rift. Used for Oculus Rift customizations and integration with the Unity game engine. Available on all major systems.

[1] Self-funded as of January 22, 2015.
[2] Free and subscription based version available.
[3] A derivation from Javascript.

- Leap Motion SDK is the source development kit for the Leap Motion device. Used for customizing the Leap Motion (custom hand gestures) and integration with the Unity game engine. Available on all major systems.

3.2 Other Concerned Systems

The purpose of this project is to demonstrate the idea of a true three dimensional real-time castle defense game. Therefore, the first iteration will be exclusively in the Unity environment and will not be concerned with other systems.

3.3 External Critical Data

The project will be built entirely from scratch. There is no existing critical data since there does not exists a castle defense game that utilizes the Leap Motion, Oculus Rift, and Unity game engine.

3.4 User Interface Guidelines

The goal is to have input purely from the Leap Motion and Oculus Rift. The Leap Motion records hand movements and gestures while the Oculus Rift provides camera rotations all of which translates into having an effect on the virtual world.

4 Functional Requirements

4.1 User Interface Finalization

User interface specifications are subject to change based on prototyping. A proof of concept for specific interface controls, usability, and organization will be necessary in determining the final details. This project will be following Agile development and iterative development methodologies for adaptive planning.

4.2 Detailed Specifications

4.3 Functional Testing

Correctness of program will be verified through automated testing and QA testing. Build pipeline will consist of unit testing, integration testing, and acceptance testing.

5 Non-Functional Requirements

5.1 Security

There are no runtime security requirements for this project at the moment.

5.2 Fault Tolerance

Main area of concerns are incomparability issues in versions between Leap Motion, Oculus Rift, and Unity game engine. Because the goal of the project is to demonstrate an idea, for simplicity

and development purposes, the versions of all technology will be restricted to the following until further notice:

Leap Motion SDK	2.2.2.24469
Oculus Rift SDK	0.4.4 Beta
Unity	4.6.1

Another issue may be software and hardware crashes. This is not a critical issue since there is no need for data recording. Therefore, the magnitude of this issue is the same as every other application on a computer.

5.3 Performance Requirements

For a majority of games, frames-per-second (FPS) is a typically benchmark used to measure the performance of a game. Performance is typically tiered as follows:

Frames-Per-Second	Performance
< 30	Limited
$30 - 40$	Average
$40 - 60$	Good
> 60	Optimal

The Oculus Rift, however, supports up to 70 FPS, and it is strongly recommended for developers to ensure that their games run at such a performance in order to prevent discomfort from experiencing a choppy virtual world. As such, 70 FPS will be our performance goal.

5.4 Scalability/Throughput Requirements

For simplicity in demonstrating true real-time gameplay, the project will utilize the minimum number of devices that communicate directly via LAN.

5.5 Non-Functional Testing

All non-functional requirements, as specified above, can be verified by user run-through of the game.

6 Process Requirements

6.1 What Must be Done When?

6.1.1 What will be Delivered, and to Whom?

6.1.2 In What Form will it be Delivered?

6.2 Who Will Sign Off When it is Done?

7 Summary

8 Signatures

Team Representative

_____

Name (Printed)

_____

Signature

_____

Date