Virtual Reality Dashboard

Software Requirements Specification

Version 1.1

10/19/2015

Justis Gill

Trevor Vasher

Joshua Marshall

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 10/4/2015 | Version 0.1 | Justis Gill | Setting up the document |
| 10/6/2015 | Version 0.5 | All | Filling out information |
| 10/11/2015 | Version 1.0 | All | Final revisions |
| 10/19/2015 | Version 1.1 | Justis, Josh | Small changes, updated DFD |

# **Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |
| --- | --- | --- |
| **Printed Name** | **Title** | **Date** |
| Justis Gill | Project Lead | 10/11/2015 |
| Trevor Vasher | Project Member | 10/11/2015 |
| Joshua Marshall | Project Member | 10/11/2015 |

**Table of Contents**

**REVISION HISTORY**

**DOCUMENT APPROVAL**

**1. INTRODUCTION**

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, and Abbreviations

1.4 References

1.5 Overview

**2. GENERAL DESCRIPTION**

2.1 Product Perspective

2.2 Product Functions

2.3 User Characteristics

2.4 General Constraints

2.5 Assumptions and Dependencies

**3. SPECIFIC REQUIREMENTS**

3.1 External Interface Requirements

*3.1.1 User Interfaces*

*3.1.1.1 Map Visualization*

*3.1.1.1.1 State Details*

*3.1.1.2 Timeline Animation*

*3.1.2 Hardware Interfaces*

*3.1.3 Software Interfaces*

*3.1.4 Communications Interfaces*

3.2 Functional Requirements

*3.2.1 Map Visualization*

*3.2.1.1 Map*

*3.2.1.2 Timeline Animation*

*2.3.1.3 Control Scheme*

*3.2.1.4 User States*

*3.2.1.5 Filtering*

*3.2.2 Roller Coaster Visualization*

*3.2.2.1 Perspective & Concept*

*3.2.2.2 Graph Inlay*

*3.2.2.3 Event Markers*

*3.2.2.4 Control Scheme*

*3.2.3 Program is compatible with Oculus Rift*

*3.2.4 Program capable of visualizing properly formatted datasets*

3.3 Non-Functional Requirements

*3.3.1 Performance*

*3.3.2 Reliability*

*3.3.3 Availability*

*3.3.4 Maintainability*

*3.3.5 Portability*

3.4 Design Constraints

3.5 Logical Database Constraints

**4. Analysis Models**

4.1 Dataflow Diagram (DFD)

# **1. Introduction**

This introduction to the Software Requirement Specification (SRS) document provides an overview of the complete SRS document.

## **1.1 Purpose**

The purpose of this document is to give a detailed description of the requirements for the “Virtual Reality Dashboard” software. It will show the intended features and documentation on how they work, giving a preliminary look at how the user will interact with the software. This document is meant to be both a proposal for the client and a reference for developing the first version of the program.

## **1.2 Scope**

The "Virtual Reality Dashboard" is a Windows application which shows the users visualizations of political data in a three dimensional environment with the Oculus Rift. The application will be free and open source under a Creative Commons Attribution NonCommercial ShareAlike 3.0 license. The application will display voting information for each candidate pulled from the Huffington Post Pollster API, displaying it in formats that show specifically the ratings of each candidate over time and in comparison to the other candidates. The user will be able to interact with these visualizations, filtering both by time and candidate.

## **1.3 Definitions, Acronyms, and Abbreviations**

User - The person interacting with the data set.

Unity - The game engine used to program the system.

Oculus Rift - The virtual reality headset used to display the data.

DualShock 2 - The controller used to provide input to the program.

Huffington Post Pollster API - The source of the data, amalgamated from various polling sources.

Candidate - The people who are voted for.

## **1.4 References**

Oculus Rift System Specifications - https://www.oculus.com/en-us/blog/powering-the-rift/

Huffington Post Pollster API - http://elections.huffingtonpost.com/pollster/api

## **1.5 Overview**

The remainder of this document consists of three chapters. The second chapter provides an overview of the functionality of the system and its interactions. This chapter also introduces the constraints upon the system and assumptions about how it will run, as well as the stakeholders involved. The third chapter provides the requirements specifications in detail and a description of the interfaces used. It will break down the different areas of the program and the database requirements for proper usage. The fourth chapter highlights our analysis models, showing the data flow diagram for the overarching design and flow of the project.

# **2. General Description**

The section will give an overview of how the system works. It will be explained in the context of the application, and introduce how the functionality of it operates. It will also highlight what stakeholders will use the system and how they can use it to fit their needs. Finally, the constraints and assumptions for proper usage of the system will be presented.

## **2.1 Product Perspective**

The system consists of a Windows based application that runs on a user's computer and is displayed via the Oculus Rift. It will be used to explore the data in three dimensions, with the user interacting via a DualShock 2 controller. The user will wear the Oculus Rift with the headtracking sensor placed on their monitor to track their head movements. The application will not need to connect to outside sources and all elements are fully encapsulated within the program. The data, pulled from the Pollster API, will be parsed into Unity and automatically processed by the system for internal usage. This data consists of the location, the timeframe, the candidates, and their polling rates.

## **2.2 Product Functions**

The user will be able to explore the virtual environment created by the system. They will be able to look around by moving their head, tracked by the Oculus Rift's camera sensor. They will be able to move around and interact with the visualizations with the DualShock 2 controller. The visualizations consist of a map of the United States and a three dimensional line graph. These visualizations will be populated with the data pulled from the Pollster API.

The map will have, on each of the different states, a pin with spheres that represent the candidates, with their size defined by their vote percentage. The user will be able to filter which candidates are displayed as well as the timeframe that is shown. The three dimensional line graph will present the user with a ride along view, where they will glide along the lines representing the candidates. Signs will pop up along the way giving contextual information about the candidate around each change in percentage. They will have the ability to change which candidate's line they are riding, as well as look to the side to see how they other candidates are faring.

Both visualizations are time based, and the user will be able to pause the animation as well as move forward and backward along the time represented by the data.

## **2.3 User Characteristics**

There are two types of users that will interact the system: users who are using the prebuilt system, and users who plan on using it to display their own data. Each of these users has a different application of the system and thus each has their own requirements.

The users who use the prebuilt system will be able to explore the system, viewing and manipulating the data. They will be able to control what is displayed and will have the ability to save images of it for use outside of the system. They will interact solely with the finished product.

The users who use the system to display their own data will work with the framework created for the system to change what is displayed. They will be able to provide their own data, formatted in the specified manner, and use this to customize what is displayed within the visualizations. They can then join the users who use it for viewing, as well as distribute their datasets for users of the first category.

## **2.4 General Constraints**

The system is constrained by dependence on three dimensional graphics rendering and usage of an Oculus Rift and external controller. They must have a Windows computer with a graphics card that meets the specifications listed below, and they must have access to a working Oculus Rift Dev Kit 2 device or newer. The processor and memory must also meet the listed requirements.

Because different components will have different capabilities, users may have slightly different experiences, with higher end hardware running the system smoother, however the system is designed to meet the minimum specifications listed here. The system is designed for usage with a DualShock 2, however it will have capabilities to work with other controllers if the user does not have access to this.

For users who want to provide their own data, they must format their data in the format listed in this document for it to be properly parsed and displayed within the visualizations.

## **2.5 Assumptions and Dependencies**

One assumption that must be made is that the user has the necessary hardware to run the Oculus Rift. If there hardware is not compatible, the program might not run as intended or at all. Another potential issue would be the user attempting to include a data set that is nor formatted properly, which may lead to bizarre results. We can account for this by giving the user strict definitions of the formatting required. Beyond this, provided their hardware is functional, there should be no other issues.

# **3. Specific Requirements**

## **3.1 External Interface Requirements**

### **3.1.1 User Interfaces**

3.1.1.1 **Map Visualization**

The user should see the map at the start of the program. The user should see a black controls overlay at the bottom of the screen, centered horizontally. The controls overlay will contain the available buttons and their corresponding actions directly to the right of them.

Just above the controls overlay at the bottom of the screen the user will see a black playback timeline overlay, centered horizontally. The playback timeline overlay will consist of a bar and a position marker. The position marker will be placed according to the user’s position in the timeline. The portion of the bar to the left of the position marker will be colored red, while the rest of the bar will be black. When the timeline passes the date of a new poll, there will be a flash to indicate the new poll data.

3.1.1.1.1 **State Details**

If a state is selected, the user should see a state details overlay on the right side of the screen. The state details overlay will contain numerical and textual data about the selected state at the current point in time on the timeline. This data, from top to bottom, consists of the:

1. state name
2. the date of the position in the timeline
3. the number of observations
4. each of the candidates’ number of votes with percentages in parentheses

Upon pressing the cancel button this overlay will disappear.

3.1.1.2 **Timeline Animation**

The user should see an empty timeline at the start of the program. The user should see a black controls overlay at the bottom of the screen, centered horizontally. The controls overlay will contain the available buttons and their corresponding actions directly to the right of them.

Just above the controls overlay at the bottom of the screen the user will see a black playback timeline overlay, centered horizontally. The playback timeline overlay will consist of a bar and a position marker. The the position marker will be placed according to the user’s position in the timeline. The portion of the bar to the left of the position marker will be colored red, while the rest of the bar will be black.

### **3.1.2 Hardware Interfaces**

### This program requires the use of an Oculus Rift for visual display and camera control. The program also uses a Playstation DualShock 2 to supplement user input. These devices will need to be plugged into a computer with at least 3 available USB 2.0 compatible ports.

### **3.1.3 Software Interfaces**

### Windows 7 or Higher is required for the program to run. This machine must also install version 0.7 of the Oculus Rift Runtime to provide a driver interface to the Oculus Rift.

### **3.1.4 Communications Interfaces**

There is no web communication necessary, as the input dataset is expected to be static. The only special communication that will take place reading the dataset from a file. The rest of the communications are the typical operating system communications with the drivers and runtime needed to run and display the program.

## **3.2 Functional Requirements**

### **3.2.1** **Map Vi**sualization

3.2.1.1 **Map**

There will be a 2D map of the United States. Three dimensional pins with several spheres on each pin will be placed in each state that has provided with polling data for the relevant election. The spheres will be color coded to an on-screen legend labeled with candidates names. Each color coded sphere will represent that candidates popularity in the state which that sphere’s pin resides. The spheres will be on the pin in order of their popularity with the most winning candidate’s sphere on top and the losing candidates sphere on the bottom. The spheres be sized based on the candidates popularity. The larger the sphere, the more popular the candidate.

3.2.1.2 **Timeline Animation**

The map described above will visualize time-based polling data. The visualization will appear as follows:

1. The map will at first have no pins.
2. Pins will appear as polling dates for each state are reached. The spheres on the pins will grow, shrink, and rearrange themselves to the conventions described in the previous section as more polling data is accessed.
3. Notifications will appear as each state’s polling information becomes available.
4. The user can pause the animation at any time and view the current state of the polls.
5. When the timeline has reached its end the animation will stop and the final polling results can be viewed.

2.3.1.3 **Control Scheme**

1. Navigation State
   1. While in the navigation state, directional buttons on a game pad will be used to control zoom & panning.
   2. The L1 and R1 buttons will be used to control the height of the camera.
   3. The analog stick will be used to control movement.
   4. Oculus Rift head-tracking will control yaw & pitch.
   5. The L2 and R2 buttons will be used to fast-forward or rewind the animation.
   6. Start will be used to pause or resume the animation.
   7. The select button will be used to open a menu, navigated with the directional buttons, to control which candidates are displayed, allowing individual filtering or filtering by party.
   8. The x and o buttons can be used respectively to enter and exit the investigation state.
2. Investigation State
   1. There will be a large amount of data displayed on the screen in the investigation state. Navigation will be locked. The o button can be pressed to return to the navigation state.



3.2.1.4 **User States**

1. Navigation State
   1. While in the navigation state the user will be able to utilize the control scheme to navigate around the map. They will have the option of starting or stopping the animation, or entering the investigation state.
2. Investigation State
   1. While in the investigation state the user will be able to access more in depth data relating to the pin that they are investigating. This will include exact percentages, Pollster & Poll sponsor data when available.

3.2.1.5 **Filtering**

1. If polling data bas been loaded for more than one election type, e.g. Republican & Democratic primaries, the user will be able to select between the different election types and view the one of interest, or both simultaneously.

### **3.2.2** **Roller Coaster** Visualization

3.2.2.1 **Perspective & Concept**

The user will have a first-person perspective. They will be riding a line graph in a manner similar to a roller coaster. Their “track” will be the data representing the popularity of one candidate. There will be other parallel tracks alongside of the user’s track. These will represent the other candidates. Looking to the side you will be able to see how tall your track is compared to the other tracks, this represents the popularity of your candidate compared to the others.

3.2.2.2 **Graph Inlay**

The user will be able to view a standard 2D version of their line graph on the top portion of the screen. There will be a marker that shows them which portion of the graph they are currently navigating from the first person perspective.

3.2.2.3 **Event Markers**

There will be event markers that resemble road-signs to mark important moments in the data set. These will need to be manually entered and customized within Unity, but will provide a way for future contributors to add a narrative to their visualization.

3.2.2.4 **Control Scheme**

1. Navigation State
   1. Pressing start will start/stop the animation.
   2. The select button will be used to open a menu, navigated with the directional buttons, to control which candidates are displayed, allowing individual filtering or filtering by party.
   3. Pressing L1/R1 or the D-Pad will switch between coaster tracks.
   4. The L2 and R2 buttons will be used to fast-forward or rewind the animation.



### 3.2.3 Program is compatible with Oculus Rift

The program will be able to be viewed through an Oculus Rift on a computer that meets the listed minimum requirements and has Unity installed.

### 3.2.4 Program capable of visualizing properly formatted datasets

Our program will allow properly formatted CSV files to be loaded into Unity to act as the data source for the visualizations.

## **3.3 Non-Functional Requirements**

### **3.3.1 Performance**

The program should be able to run at a minimum of 40 frames per second on the Oculus Rift. The maximum time the program will take to load is 4 seconds.

### **3.3.2 Reliability**

The program should be able to reliably perform the functions listed in the functional requirements on a system that meets the minimum specifications without crashing.

As this program is an experimental project, some bugs are to be expected. If the program is being used for its intended purpose these should not be prevalent.

### **3.3.3 Availability**

The program will be distributable via a .zip file which contains the Unity project source. Means of distribution will be discussed before the conclusion of the project.

### **3.3.**4 **Maintainability**

The program will not require maintenance as is. However, it will provide a framework for future contributors to create their own visualizations. The source code will be commented and written to be easily readable by future contributors. Documentation will be provided to make the code more easily understandable.

### **3.3.**5 **Portability**

The program does need a computer with fairly high minimum specifications (listed). It will also require Unity 5 to be installed, and an Oculus Rift connected for proper viewing.

## **3.4 Design Constraints**

Because system resources are not unlimited, the number of data elements that the program will accept is 100,000. If more data is received after that limit, it will be truncated to include only the last 100,000. In addition the maximum supported resolution will be 1920x1080 pixels, in line with the maximum resolution of the Oculus Rift.

## **3.5 Logical Database Requirements**

A csv file will be used to store the polling data. The csv file will have the following fields:

1. State - State in which the poll took place
2. Pollster - Polling agency
3. End Date - End date of poll.
4. Method - Polling method
5. Source - Publication source of poll
6. Affiliation - Any party Affiliation
7. Survey Houses
8. Sponsors
9. Questions - Subject of the poll
10. VoterStats - Characteristic of some surveyed
11. VoterStatsNum - Number of those sampled with that characteristic
12. Candidates 1-8
13. Percents 1-8

Some of this information may not be used in the first version of this program, but will be available for later use.

# **4. Analysis Models**

## **4.1 Data Flow Diagrams (DFD)**

All data that will be used for the visualization will be read from a .csv file. From this data, polls and their attributes are extracted. The poll attributes be merged into display data after some processing is performed. This display data is then combined with user input to determine what is shown on the screen at a given time, and from what perspective. The user can filter the data and advance or rewind the timeline as desired.

