*VR Dashboard*

**PRODUCT DESIGN SPECIFICATION**

Version 1.5

11/4/2015

**VERSION HISTORY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 0.5 | Josh, Justis, Trevor | 10/25/2015 | All | 10/25/2015 | Initial Design Definition draft |
| 1.0 | Josh, Justis, Trevor | 11/1/2015 | All | 10/25/2015 | Final Design Document |
| 1.5 | Josh, Justis, Trevor | 11/4/2015 | All | 11/4/2015 | Updates as discussed with client |

**UP Template Version:** 12/31/07

**TABLE OF CONTENTS**

[**1** **INTRODUCTION**](#h.1fob9te)

1.1 Purpose of The Product Design Specification Document

**2** **GENERAL OVERVIEW AND DESIGN GUIDELINES/APPROACH**

2.1 Assumptions / Constraints / Standards

2.2 Data Constraints

[**3** **ARCHITECTURE DESIGN**](#h.1t3h5sf)

[3.1 Hardware Architecture](#h.2s8eyo1)

[3.2 Software Architecture](#h.17dp8vu)

[3.3 Performance](#h.lnxbz9)

[**4** **SYSTEM DESIGN**](#h.35nkun2)

[4.1 Use-Cases](#h.1ksv4uv)

[4.2 Sequence diagram](#h.44sinio)

[4.3 Data flow diagram](#h.2jxsxqh)

[4.4 Database Design](#h.z337ya)

[4.5 User Interface Design](#h.4i7ojhp)

[**5** **PRODUCT DESIGN SPECIFICATION APPROVAL**](#h.2xcytpi)

[**APPENDIX A: REFERENCES**](#h.3whwml4)

# **INTRODUCTION**

## **PURPOSE OF THE PRODUCT DESIGN SPECIFICATION DOCUMENT**

This Document will outline the design of the VR Dashboard (Data Visualization) project for the Fall Semester CSC 4996 class at Wayne State University. The general guidelines, architecture design, and system design will be discussed. The document will give the reader a basic overview of the specialized VR hardware we are using, and describe in detail the program we are creating to utilize this hardware for the task of data visualization. Its intended audience is the client and the development team.

# **GENERAL OVERVIEW AND DESIGN GUIDELINES/APPROACH**

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

## SYSTEM CONSTRAINTS

In order to use our program, the user must have a capable computer system as well as an Oculus Rift DK2.The minimum system requirements are as follows:

1. A desktop computer running Windows 7 or Windows 8,
2. 2 USB 2.0 ports (at least one powered),
3. Dedicated graphics card Nvidia GTX 600 series or AMD Radeon HD 7000 series (or better) with DVI-D or HDMI graphics output.
4. Unity 5 must be installed on the system.

## DATA CONSTRAINTS

The user will need to have a properly formatted CSV file to view data that they provide. Alternatively, one file will be included with the program for the user to view. The proper format is described in section 4.4 of this document.

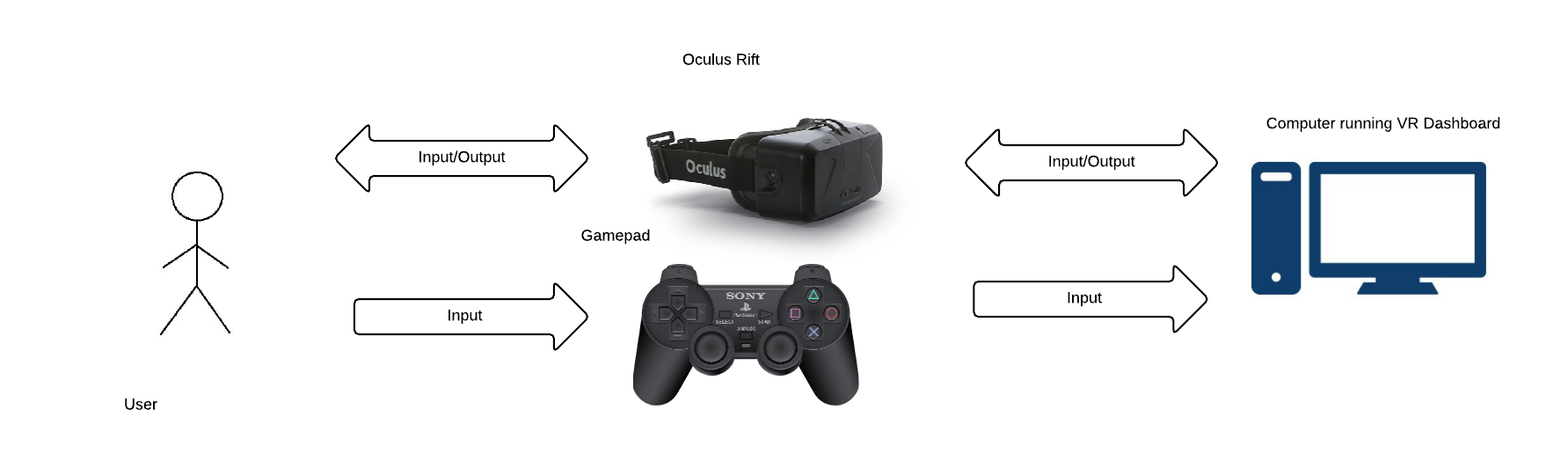
The user has the option of providing an Event Marker CSV file, the format for this is also specified in section 4.4.

# **ARCHITECTURE DESIGN**

This section outlines the system and hardware architecture design of the system that is being built. The application interacts with the Oculus Rift SDK to manage generating the proper image for virtual reality and monitor headtracking inputs.

## **HARDWARE ARCHITECTURE**

The user provides input via the Oculus Rift and the gamepad, which then send their input data to the computer running VR Dashboard. It interprets the input and then sends the corresponding output to the user via the Oculus Rift.

* + 1. Oculus Rift Overview

Oculus Rift is a Virtual Reality Headset with 7 inch LCD display. The screen is divided into two sections - 640 \* 800 pixels per eye. The User views screen through two lens cups focused on the display. The Device has built in motion & orientation sensing by mean of a gyroscope, an accelerometer and a magnetometer. This allows for an immersive VR experience that can track the movement of the users head allowing them to look around the virtual environment.

## **SOFTWARE ARCHITECTURE**

* + 1. Unity Overview

Unity is a game engine that is used to develop games for desktops, mobile devices, browsers and game consoles; it is used at the basis of our project. It provides the user with a development environment suitable for creating 2D or 3D games and applications. The user is able to create scenes, which are populated by game objects. C#, Javascript, and Boo are the languages that are used for scripting in Unity. Scripts can manipulate all aspects of the game including scenes, game objects, and other scripts. The scripts are compiled at runtime.

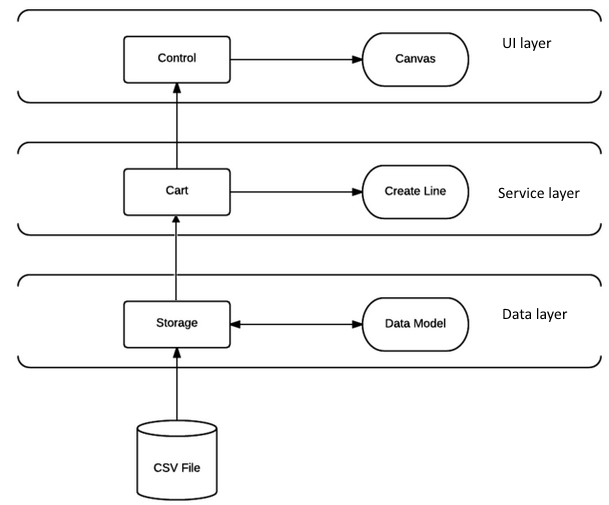
Unity is used as the basis of our program. We extend upon the functionality of Unity by creating custom C# scripts. Because the scripts are compiled at runtime, they involve just in time compiling, rather than ahead of time compiling. The Unity framework itself is compiled ahead of time.

* + 1. Main Menu

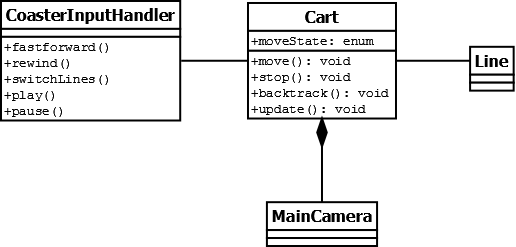
The user is presented with a main menu screen where the desired visualization and CSV file can be selected. The visualization options will be implemented as UI buttons and the CSV file selection will be implemented as a dropdown list. From this screen, the player’s headtracking will be used as a Cursor within the worldspace of the application. If the user looks at one of the UI buttons it will become highlighted, allowing the user to select it by pressing X. This will trigger a script to either load the scene for the selected visualization or activate the CSV file dropdown.

* + 1. Rollercoaster

The CSV file is loaded into storage, where it is then transformed and stored again by the data model. This is then accessed by the cart object, which uses the data to create the lines for the graph and is directly controlled by user input. This is then accessed by the UI control scripts, which adds the data to the canvas for the UI.

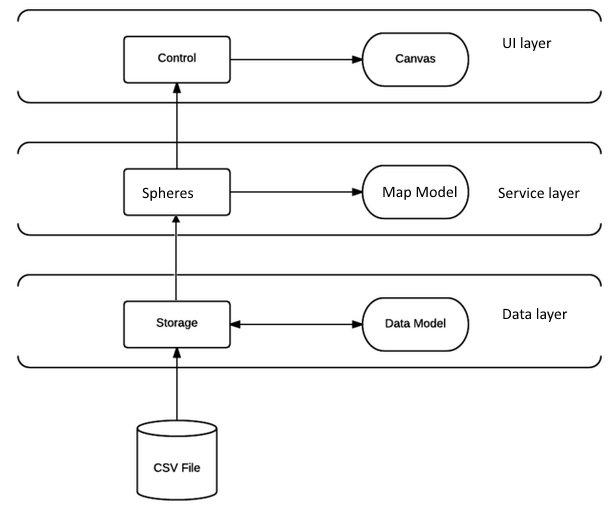


The main camera is attached to the cart, which stores its state and has functions for moving, stopping, backtracking, and updating its current position. It listens for input, which will be in the form of fast forward, which speeds up the movement, rewind, which reverses it, switch lines which changes what line you’re on, play which starts the animation, or pause which stops it. It is combined with a line object, which the cart gets data from and rides on top of.

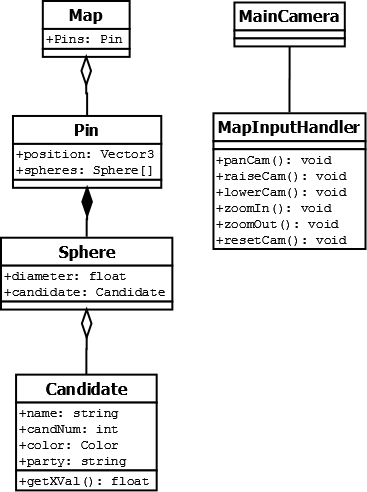


* + 1. Map

The data layer formats the data into spheres, which are then read into the map model and processed, placed where they belong on the map. The UI controller accesses the data from the spheres and map model and use this to fill in UI information on the canvas.



The map contains pins, which hold their position and the spheres that they will display. The spheres and associated with a candidate, which has a percentage, a name, a candidate number, a color, and a string for their party. Meanwhile, the input acts directly on the main camera, allowing you to pan, raise the height, lower the height, zoom in and out, and reset to initial position.



## **PERFORMANCE**

The program will perform at a minimum of 40 frames per second on a system that meets the minimum specs for the Oculus Rift. The number of visual objects in our map visualization does not increase as the data set grows, and thus the frames per second for this visualization are constant. In the coaster visualization, the number of objects grows by the number of rows present in the data set.

To limit the frame rate loss of an extremely large number of objects on the screen, a data element cap must be added. A data element cap of 100,000 is enforced. This cap will keep the frames per second from dropping below our minimum and keep the load times of the program at 4 seconds or less.

# **SYSTEM DESIGN**

## **USE-CASES**

See included Use Cases Document.

## **SEQUENCE DIAGRAM**

See included sequence diagram JPEG files.

## **DATA FLOW DIAGRAM**

See included data flow diagram PNG file.

## **DATABASE DESIGN**

* + 1. Data (CSV) Format

The CSV file should be written in plain text, with each row representing one poll. Only one CSV file can be loaded at a time, however the user will have the option to choose which CSV file they would like to use. The formatting for each row is defined to match with the Huffington Post Pollster API, however only certain categories are mandatory. Laid out the in table below is the official format from the API, however only bolded information is necessary, allowing the user to leave the other fields blank. The data is interpreted dynamically, meaning the data will search through the document for the number of polls, and as long as it is below our maximum of 100,000, incorporate all of the lines into the file. Although the visualizations are designed to show only up to 5 candidates at a time, an unlimited amount of candidates can be represented in the CSV, to be picked later. If there are multiple polls represented on the same day, first a duplicate check will be run. If the polls are found to be the same, the duplicate will be dropped. If they are found to be different, they will be merged, so that each candidate’s percentage will be recalculated based upon the number of voters in the polls to be merged. If a candidate does not appear in all of the polls, they will receive a weighted average, the same as the rest of the candidates.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Col 1** | Col 2 | **Col 3** | Col 4 | Col 5 | Col 6 | Col 7 | Col 8 |
| **State** | Polling Agency | **End Date** | Method | Source | Affiliation | Survey Houses | Sponsors |
| Col 9 | Col 10 | **Col 11** | **Col 12** | **Col 13** | **...** | **Col N-1** | **Col N** |
| Questions | Voter Stats | **Number of voters** | **Candidate 1** | **Candidate 1 Percent** | **Candidate/Percent** | **Candidate N** | **Candidate N Percent** |

The following format is for the eventMarker.csv file. This file will be used to place event markers into the coaster visualization. The first column contains the candidate, the second column contains the date, and the third column contains the text that the user wishes to be displayed on the event marker. The maximum length for the this text is 30 characters. If more characters are entered they will be cut off by the program and only the first 30 will be displayed.

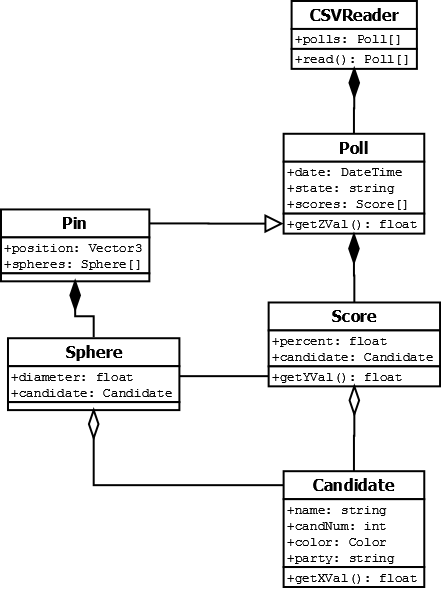
|  |  |  |
| --- | --- | --- |
| **Col 1** | **Col 2** | **Col 3** |
| **Candidate** | **Date** | **Display Text** |

* + 1. Data Processing

The CSV file is loaded into memory as a raw data string, before being converted into a filtered array, where only relevant information (poll data, state, number of pollers, candidate, percent ... candidate, percent. It is then changed into PollObjects, and then formatted as described in the diagram.

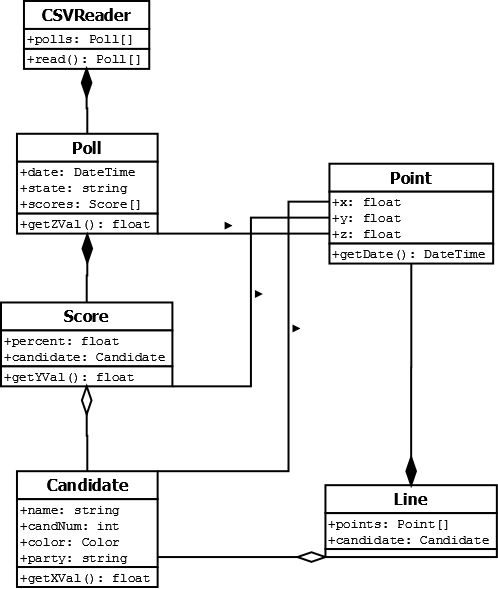
* + 1. Map Data

The map data is read in from the CSVReader, which consists of polls, representing polls within that state. These polls are used for pins, which contain spheres that model the candidates and the scores they received.



* + 1. Coaster Data

The CSV Reader creates polls, which contain scores for candidates, and are used to generate points on a line for the visualization.



## **USER INTERFACE DESIGN**

VR presents an interesting challenge to user interface design. Since the screen reacts to the user's head movement, having the UI drawn as an omnipresent screen overlay can be very disorienting. To solve this we will integrate our UI elements into the worldspace. These elements will all be children of Canvas objects.

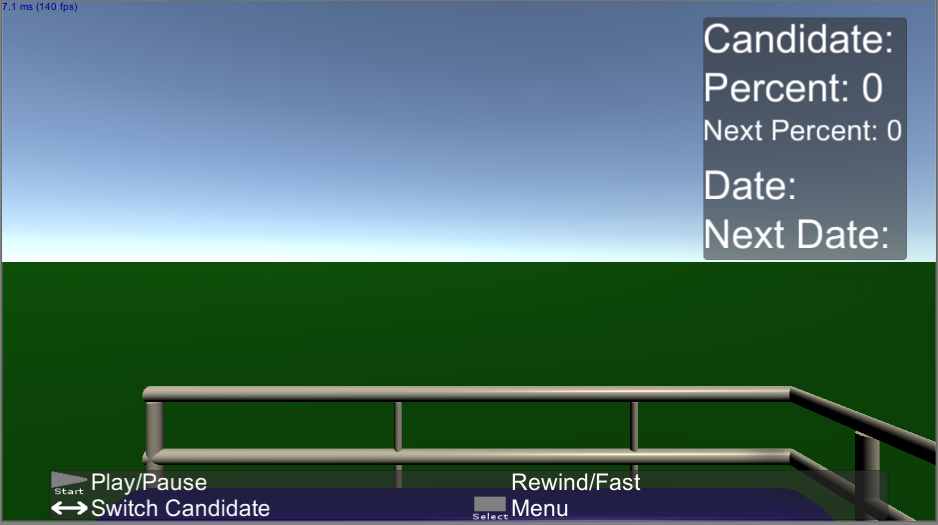
1. Main Menu

* The visualization buttons will be in the vertical center of the screen in a row.
* The CSV file dropdown will be centered just below the visualization buttons.

1. Coaster

* The candidate that each line represents will be displayed on the car.
* Each data point will have a large ring around it. These rings will contain the date and percentage of the poll the candidate held on that date, centered and on separate lines.
* Event indicators will appear as larger rings around the line. The text on these rings will also be centered.
* A 2D line graph will be displayed on a billboard in the distance oriented to the user's front-left. This graph will have position marker that represents the player's position on the graph.
* There will be a controls overlay that is drawn as a screen overlay in the center of the bottom of the screen. However this overlay can be toggled such that the user can refer to it only when needed.

\* Mockup of Coaster UI

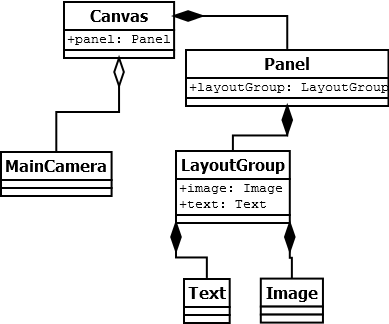


1. Map

* There will be a controls overlay that is drawn as a screen overlay in the center of the bottom of the screen. However this overlay can be toggled such that the user can refer to it only when needed.

\* This mockup is pending updates from recent requirement changes





# **PRODUCT DESIGN SPECIFICATION APPROVAL**

The undersigned acknowledge they have reviewed the VR Dashboard **Product Design Specification** document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | Justis Gill | Date: | 11/4/2015 |
| Print Name: | Justis Gill |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | Trevor Vasher | Date: | 11/4/2015 |
| Print Name: | Trevor Vasher |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Signature: | Joshua Marshall | Date: | 11/4/2015 |
| Print Name: | Joshua Marshall |  |  |

**Appendix A: References**

The following table summarizes the documents referenced in this document.

|  |  |  |
| --- | --- | --- |
| **Document Name and Version** | **Description** | **Location** |
| Use Cases 1.0 | Use Cases for our project | Use Cases.docx |
| Sequence Diagrams 1.0 | Sequence Diagrams for our project | /Sequence/ |
| DFD 1.0 | Data Flow Diagram for our project | data\_flow\_diagram.png |