CS CAPSTONE REQUIREMENTS DOCUMENT

OCTOBER 29, 2019

DESIGN AND IMPLEMENTATION OF A FRAMEWORK FOR BIO-INFORMED 3D USER INTERACTION

PREPARED FOR

OREGON STATE UNIVERSITY

RAFFAELE DE AMICIS

PREPARED BY

GROUP 51

BIOMR

LEY ALDINGER

AYUSH CHOUDHURY

KYLE HIEBEL

ZUNYUE QIU

Abstract

In order to improve the experience of extended VR sessions, we intend to study the effects of specific environmental factors on the time perception of VR users. This study requires a system architecture that allows a researcher to gain insight on which environmental factors affect a VR user's perception of time. Many components in this architecture already exist, such as the sensors, game engine, and head mounted display. This project requires the design and implementation of an API which manages communication between biometric sensors and a VR environment. In addition, this API must give researchers the ability to modify the VR environment in real-time based on sensor data and the current state of the user in VR. The API needs to maintain a data storage to keep track of sensor data and VR activities. This will allow researchers to correlate the biometric data with the user's environment, giving insight into the effects of VR on the human body.

CONTENTS

1 Introduction 2

1.1 Purpose . . . 2
1.2 Scope . . . 2
1.3 Glossary . . . 2
1.4 Overview . . . 2
1.4.1 Product Perspective . . . 2
1.4.2 Product Functions . . . 3
1.4.3 User Characteristics . . . 3
1.4.4 Limitations . . . 3

2 System Requirements 3

2.1 High Level Requirements . . . 3
2.2 Functional Requirements . . . 3
2.3 Usability Requirements . . . 5
2.4 Performance Requirements . . . 5
2.5 System Interfaces . . . 5
2.5.1 User Interfaces . . . 5
2.5.2 Software Interfaces . . . 5
2.5.3 Hardware Interfaces . . . 6

3 Verification 6

3.1 Functional Testing . . . 6
3.2 Usability Testing . . . 6
3.3 Performance Testing . . . 6
3.4 Interface Testing . . . 6

4 Appendices 6

4.1 Constraints . . . 6
4.2 Assumptions and Dependencies . . . 7

5 Timeline 7

5.1 Gantt Chart . . . 7

1 INTRODUCTION

1.1 Purpose

As virtual reality technology becomes more popular, users are spending increasing amounts of time in virtual worlds. In the near future, some on-site jobs will be replaced with cyber-physical jobs which place the worker in a virtual environment for 8 or more hours. Due to this impending upsurge in the use of virtual reality, it is increasingly important to understand the effects which being in a virtual environment may have on the body. Understanding the physiological effects of VR will enable developers to create virtual reality software designed for optimal user experience, and allows for the development of systems which adapt in response to user physiological data in near real time.

1.2 Scope

The project will entail the research, design, and implementation of a system architecture which allows for communication between biometric sensors and a game engine. The sensors, virtual reality headset, and game engine are components which will be provided by the client. The individual components will be chosen based on their price and functionality. A custom API will be developed to transmit data between the sensors and game engine. This custom API does not need to be developed from scratch, and will likely be built as an extension of an existing API. Finally, a virtual reality application will be developed in a game engine to validate the system architecture. Again, this VR environment can be created by modifying an existing scene to speed up development.

1.3 Glossary

- AR - Augmented Reality
- API - Application Programming Interface
- ECG - Electrocardiogram
- EMG - Electromyography, an electrical diagnostic medical technique used to assess and record electrical activity produced by skeletal muscles.
- GSR - Galvanic skin response
- HMD - Head Mounted Display
- HTC VIVE - Virtual reality headset. Released 2016.
- Oculus Quest - Virtual reality headset. Released 2019.
- SDK - Software Development Kit
- System Architecture - A conceptual model that defines the structure, behavior, and more views of the system.
- UE4 - Unreal Engine 4 (Game Engine)
- Unity - Game Engine
- VR - Virtual Reality
- XR - Cross Reality. Virtual and Augmented Reality combined.

1.4 Overview

1.4.1 Product Perspective

The product is a system architecture that will allow researchers to investigate how a VR environment affects a user's perception of time. The system architecture consists of many hardware and software components that must be linked together to be fully functional.

1.4.2 Product Functions

The product needs to have a few main functions. First, an API must facilitate communication between biometric sensors and a VR environment. This API must have the ability to alter the VR environment in real-time based on the sensor readouts. Data from sensors will be correlated with actions of the user in VR to complete the research study.

1.4.3 User Characteristics

Researchers will use the system architecture to run tests on individual subjects. The API will contain an interface allowing the researcher to set triggers based on sensor levels which will modify the VR environment. The subject in VR will experience these changes in real time, and their subsequent biometric sensor data will be recorded.

1.4.4 Limitations

This project will be based largely off existing technologies which may limit the capability of the system. Some hardware devices, such as an HTC Vive, ECG, and EMG sensor will not be modified within the scope of this project. Additionally, sensor firmware and the game engines themselves will be used "as is" throughout the course of the project.

2 SYSTEM REQUIREMENTS

2.1 High Level Requirements

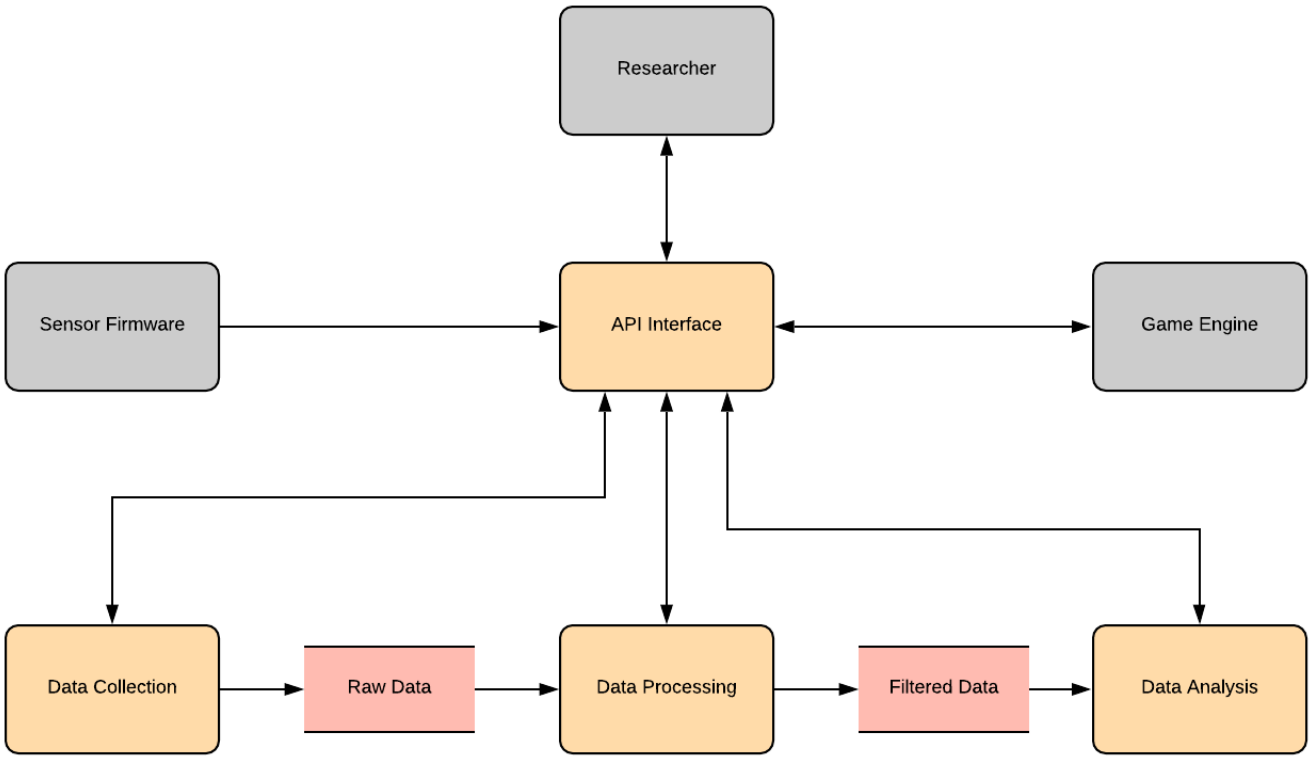
Fig. 1. The system architecture broken down into it's 5 major components. The API will be developed in-house while most other components will be purchased.

The system architecture can be broken down into 5 major components. First, the sensors collect biometric data from the user's body. The sensor's firmware reads raw data from the sensors and makes it accessible to other components. The API must be able to connect to an arbitrary sensor's firmware and read raw data. The API will need to collect, analyze, and process this sensor data. Collecting data entails reading data from sensors or the game engine and storing it in a file or database. Analyzing includes filtering the data to show only useful entries. Processing data involves extrapolating meaning from the useful entries. The game engine will need to both read data from the API, and provide information about the user's state in VR to the API. Lastly, the Head Mounted Display (HMD) must respond to changes in the game engine, such as user movement or object interactions.

2.2 Functional Requirements

The system architecture must allow a researcher to gain insight on which environmental factors affect a VR user's perception of time. There needs to be some quantitative measurement of time perception in the system architecture, as asking subjects to interpret their own time perception introduces a great deal of error.

Fig. 2. A breakdown of the data flow within the API. All data must flow through the API Interface to any external component.



The main component of the system architecture which needs to be developed is the API. This API is a set of functions contained in an interface which a user may include in their application. The API will operate as middleware between the sensors and game engine. This interface will act as a query manager, allowing the researcher, game engine, and sensor firmware to read and write information to a data store.

The API must have the ability to collect raw data, meaning it must read, then persistently store the data. The data must be stored in memory or in a file, depending on size, and can be accessed later. Data collection must include a timestamp so that entries from the game engine can be correlated with entries from the sensors. The reading of data must be independent of the sensor type and game engine, allowing the system architecture to change while maintaining the same API.

Once the data is written, it must be processed to filter out unnecessary entries, thus improving information density. These filters are to be designated by a researcher. The filtered data should be analyzed by correlating VR events with sensor data.

The game engine must be able to send signals to the API indicating what activity the user is performing. These activities include navigation, item selection, item manipulation, and dynamics. When the game engine indicates that the user is performing one of these activities, the API will timestamp the activity and store it alongside the sensor data.

The game engine must also be able to receive signals from the API indicating a scene modification. These modifications must be coded into the test scenes in UE4 or Unity. The scene must be able to modify the light conditions, weather conditions, and environmental dynamics (object movement) based on which signal is sent from the API.

The product must allow a researcher to retrieve biometric sensor data logs encapsulating the physiological user

experience over the course of a virtual experience. Data needs to be accurate. The solution should come as a package which will be obtainable and usable by developers. Data should be transmitted quickly enough to be used in an application, ie. in near-real time.

2.3 Usability Requirements

There are two classifications of users for this system, researchers and subjects. There must be a simple 2D GUI for researchers to interact with the API, either to read data or set triggers. This GUI must be simple and intuitive such that researchers not familiar with the project can use the interface.

Subjects are the people who will be in VR environments and have biometric sensors reading physiological data about them. Subjects must not be shown any 2D interfaces or instructions. They must be immersed in a 3D VR experience which is intuitive enough that no text instructions are required. Instructions can be considered an environmental factor and need to be removed to ensure the accuracy of the experiment.

2.4 Performance Requirements

The performance requirements vary depending on the component of the system. The system as a whole must operate in near real-time, which is loosely defined as low enough latency that the delay is not perceivable to a user. Certain components, such as the game engine, need to operate more quickly because they need to push 90 frames per second to the head mounted display. On the other hand, the API can take multiple frames to update the environment in the VR scene without causing discomfort to the user. The API typically needs to respond in roughly 100ms because its contribution to the scene is not completely essential. Since the API is collecting physiological data from the user, the system architecture needs a high storage capacity. Usability is also important to our performance requirements, concerning availability, interoperability, safety, efficiency and flexibility, which are essential quality attributes for the VR environment.

2.5 System Interfaces

2.5.1 User Interfaces

Users will interface with the product through an API, or set of functions that come included in the package. These functions will allow the user to query information about the sensor state and read processed data. Additionally, there must be an internal observer script which will continually track the incoming data. Within this script, a researcher can dene certain triggers which can automatically modify the virtual environment when sensor readouts reach a certain threshold. This is useful because the script can experimentally test multiple levels of environmental factors in real-time without input from a researcher.

2.5.2 Software Interfaces

The API must have a one-way interface with sensor firmware. The firmware will either push raw data to the API or allow data queries, upon which the data will be read and stored. The API must also have a two-way interface with the game engine. The game engine will report activities being performed by the user, and the API will provide the game engine with sensor levels. While this exchange of information is happening, the API will record and track patterns between sensor data and data from the game engine.

2.5.3 Hardware Interfaces

The hardware interfaces include biometric sensors and VR HMDs. The sensors will need to be researched and purchased, and will have a firmware interface for data gathering. The chosen sensors must be able to provide raw data through their firmware to the API, either as a query or a push. The HMD must have a 2-way interface with the game engine to allow player movement and actions to modify the camera position. The firmware on the sensors and HMD will already be developed, so correctly chosen components should require no further work.

3 VERIFICATION

3.1 Functional Testing

To test the accuracy of the data collected and processed, unit tests will be created for each type of data processed by the API. These tests will compare the values that were output by the framework's data processing section, and compare them to expected values. The tests will be conducted for both game engines for each sensor to ensure every part of the project is functional.

3.2 Usability Testing

An application will need to be created as a proof of concept that the developed framework can be used as an API for VR application development. A user study will be conducted to test how usable the developed framework is from the perspective of a VR application user. This will test if the framework truly allows a VR application to adapt to the user's needs in a perceptible way.

3.3 Performance Testing

There are two main ways to test the performance of the solution. The first is to test the framerate of the VR application using the solution and see if it the API interface has a significant impact it. The next is to look at the latency between data updates. This will be measured by timing how long it takes for new data to be sent from the sensor to the API and for it to be processed by the framework. The performance can also be measured subjectively during the user study.

3.4 Interface Testing

The user interfaces, or API calls that a developer would use, will be tested similarly to the functional requirements. Unit tests will determine whether the calls produce the results expected for given inputs. The software interfaces will also be tested with unit tests to ensure the framework is correctly processing the raw data it receives. Much of the interface verification will be accounted for during functional testing. Since the design of the hardware interfaces are outside of the scope of this project, no testing will be done to ensure their correctness.

4 APPENDICES

4.1 Constraints

The choice of game engine is limited to either Unity or Unreal Engine 4. This constraint is in place because team members have competence in the use of these two engines. It is more time and cost effective to use game engines with previous knowledge. Additionally, Unity and Unreal Engine are popular tools used commonly for commercial VR development, and therefore are representative of the tools used by our target user population.

Another constraint are the limitations of the biometric sensors themselves. The project will be limited by the accuracy and type of data received from these sensors. Furthermore, the project will be limited by the speed of data transfer either by wireless internet or a hardwired connection.

4.2 Assumptions and Dependencies

The system architecture is dependent on many hardware components, which are assumed to operate as expected. These hardware components include the biometric sensors, head mounted displays, and a VR capable PC. The software must compile and run using this specific configuration and be able to interface with the multiple external hardware components.

In addition, it is assumed that the biometric data from the sensors is calibrated and accurate. The API will guarantee that the sensor data gathered will be the same information sent to the VR application. This makes the effectiveness of the system dependent on the accuracy of the sensors.

5 TIMELINE

5.1 Gantt Chart

Fig. 3. A timeline of when certain requirements must be met. The career fair is May 15, 2020.