Oregon State University
College of Engineering

CS Capstone Progress Report

February 28, 2020

Design and implementation of a framework for bio-informed 3D user interaction

Prepared for

Oregon State University College of Engineering

Raffaele de Amicis _____ _____
Signature *Date*

Prepared by

Group 51
BioMR

Ley Aldinger _____ _____
Signature *Date*

Ayush Choudhury _____ _____
Signature *Date*

Kyle Hiebel _____ _____
Signature *Date*

Zunyue Qiu _____ _____
Signature *Date*

Abstract

Our project consists of creating a system architecture which will enable a researcher to study how certain environmental parameters affect a subject's perception of time in VR. So far, we have made progress on a test scene, acquired hardware, and read research papers to help the design process. The team had to solve many problems involving interfacing between a custom API and a VR environment. Overall, team members have done great work this term, but need to improve communication next term so the development process goes smoothly.

CONTENTS

1 Project Overview

Over the last few years, virtual reality (VR) and other immersive technologies have rapidly grown in popularity. A new usage for this technology is emerging in manufacturing shops throughout America. Smart machines, with enhanced visualization capabilities, allow for remote operation through a virtual interface. Workers using these new interfaces can expect to be in a virtual environment for up to 8 hours a day. Due to this large amount of time spent in VR, our customer would like to study the effects of certain environmental conditions on VR users, and use this information to improve VR working environments.

To gain insight on how to improve VR work environments, our customer wants to study time perception in VR. Particularly, he wants to study how lighting, weather, movement, and sound can impact an individual's time perception. The customer intends to measure time perception using biometric sensors such as eye tracking, galvanic skin response, and electrocardiogram. The insight gained will be used to optimize VR work environments to improve development costs, time-to-market, employee safety, and general efficiency for an industrialized company.

To allow a researcher to gain insight on how environmental factors affect a VR user's perception of time, we will develop a system architecture in which a researcher can perform such a study. The researcher will interact with our system, named BioMR, through a custom API which will serve as middleware between biometric sensors and a VR scene. A subject will interact with the VR environment through a HTC Vive headset while biometric sensors will quantitatively measure the subject's perception of time.

The API will provide the researcher a user interface to control and monitor the study. Using the interface, the researcher can modify effects in the game engine using value sliders. Additionally, the researcher can view data from the biometric sensors in a table. The data will be filterable and allow for custom SQL queries to find precise data entries.

Next, we will create two test scenes, one in Unreal Engine 4 (UE4) and another in Unity to validate that the system architecture is collecting meaningful data. Each of these scenes will contain controllable lighting effects, weather effects, and dynamic objects which can be modified by the researcher. Testing with two different game engines ensures that the API will be written in a portable way.

2 Project Progress

2.1 Unreal Engine 4 Scene

We found an immersive Unreal Engine 4 scene which we can use to test the system architecture. The scene is a free example from Epic Games called "A Boy and His Kite". We have downloaded the source code and explored the available space, choosing the location in Figure 1 as the player's starting position. The remainder of the Unreal Engine 4 components will be added to this base scene. Starting with an existing scene will save time compared to developing from scratch.

2.2 Unreal Engine 4 Weather Effect

We created a rain particle effect using Unreal Engine 4 (see Figure 2). This rain effect exposes two parameters which can be modified by the researcher. The size of the raindrops and the quantity of raindrops can be changed in realtime without restarting the particle system. We also implemented this particle effect to follow the camera position, so when the effect is enabled, it will always be visible to the player.

Fig. 1: This location is a good starting location because it is close to water, deer and trees. These objects can be used to influence interaction and navigation.

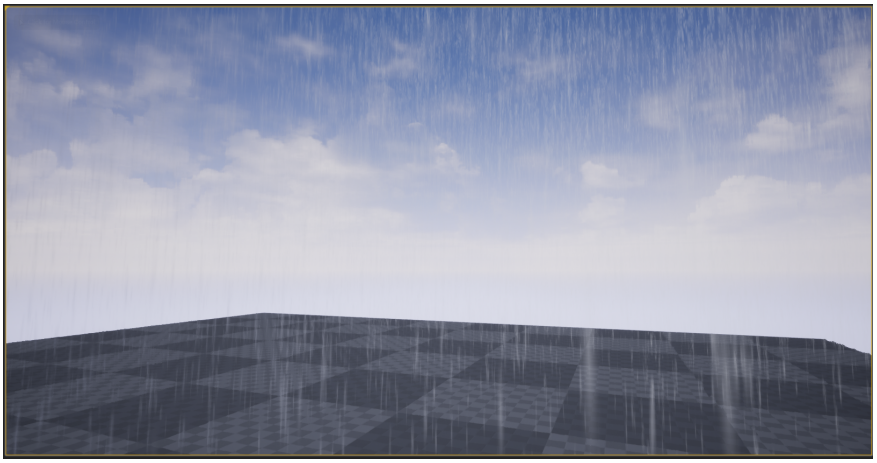

Fig. 2: Rain particles with differing sizes and starting positions.

2.3 Unreal Engine 4 Lighting

We found a library for lighting effects in Unreal Engine 4 called Good Sky. We set up the library so that the lighting changes based on a day-night cycle. There is a single parameter exposed, called time of day, which indicates where in the 24 hour period we currently are. The researcher can modify the rate at which this parameter is incremented to modify the passage of time in game. We have implemented this library in a test scene (Fig 3), but not yet in the "A Boy and His Kite" scene.

Fig. 3: Lighting effects at 6:00pm in the day-night cycle.

2.4 Unity Engine Scene

Similarly to in Unreal Engine 4 we found a suitable scene that would allow for interactions used to measure the time perception in the VR application user. The scene is called “Book of the Dead” and was developed by Unity technologies in 2018. We have tested the scene in the Unity engine to confirm that it can be used for the purposes of this project, and have briefly looked at the assets and source code to see what features the scene offers. The scene is mostly forested with trees, foliage and wind. Assets such as a vr camera can be added to the scene to allow for the interactivity needed for the project.

2.5 Unity Engine Weather Effects

While the scene we will be using already has wind, additional weather and lighting effects will also be added. We found a rain package, a lightning bolt asset, as well as skybox assets. All of these could be used to dynamically change the weather and time of day. Since there is already lighting in the scene it will also need to be manipulated dynamically by our project.

2.6 PC Acquisition

Working with the customer, we acquired a test PC for this project. The customer used his research budget to purchase a very powerful PC with a Titan RTX and i7 9900k. This system will be used for development and testing of the VR scenes. The system is located in the Computer Graphics and Visualization lab and has an HTC Vive connected to it.

2.7 Sensor Acquisition

Again, working with the customer, we have acquired a set of biometric sensors. First, we have eye tracking built into the HTC Vive VR headset. We also have an electrocardiogram to measure the electrical activity of the heart beat. Finally, we have a galvanic skin response sensor which measures the electrical resistance of the skin, influenced by emotional stress. These sensors are part of the iMotions biometric sensor suite, with associated tools for data collection and analysis.

2.8 Research Papers

We have read multiple research papers regarding time perception, attention, and focus:

- K. Lontz, "Time perception during retrospective and prospective paradigms with a distraction," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 59, pp. 1367–1371, 2013.
- B. Wandell and S. Thomas, "Foundations of vision," *PsycCRITIQUES*, vol. 42, July 1997.
- D. Vernon, "Alpha neurofeedback training for performance enhancement: Reviewing the methodology," *Journal of Neurotherapy*, vol. 13, pp. 214–227, October 2009.
- A. M. Berger and E. J. Davelaar, "Frontal alpha oscillations and attentional control: A virtual reality neurofeedback study," *Neuroscience*, vol. 378, pp. 189–197, May 2018.

These papers presented important research for our project. First, we learned that distraction can cause people to experience time more quickly. Through patterns in eye tracking, it is possible to determine a subject's level of focus. Measuring attention will allow for a quantitative assessment of time perception.

3 Problems and Solutions

3.1 Code Library

We did not know how to include common code in both Unreal Engine 4 and Unity. This is a difficult problem because Unreal Engine 4 uses C++ but Unity uses C# and Javascript. We have to write the code in a way that it can be run in both interfaces and languages. The solution is to create a C++ library for our API, and export it as a dynamic linked library (DLL). This is simple in the Unreal Engine 4 case because all code is C++. For Unity, it is more complicated because there are mixed languages, but Unity natively supports C++ DLL linking. All we have to do is create a plugin from the DLL, and Unity will allow us to call the required functions.

3.2 VR Movement

The default camera mode in the downloaded VR scene is a drone camera. This camera allows the user to move in any direction, including up and down. While wearing a VR headset, moving in any direction other than the direction the headset is facing can make a user very nauseous. To solve this, we researched different movement schemes and decided to use teleportation as our main scheme. Teleportation allows the user to pick a new location, then the screen goes blank during the movement so they do not get nauseous. This limits the user to moving along the ground, but air navigation was not a requirement for this project.

3.3 Time Perception

We did not know how to quantitatively measure time perception. We solved this by asking the customer, who is a professor, to send us some research papers. He sent four papers which helped us come to the conclusion that attention is correlated with time perception. The papers also described how attention can be measured using sensors we have. For example, we can use the eye tracker to determine how long the user has been examining a single object. Additionally, we can use eye tracking and galvanic skin response data to calculate the user cognitive load, which is linked to user time perception.

3.4 API Communication

Designing the API to communicate with the game engines was a difficult task. The problem was that each game engine operates differently and the API needs to be able to accommodate an arbitrary game engine. In order to solve this, we decided to use callback functions which can be registered with the API and called upon a certain trigger. When connecting to the API, the game engine will pass a group of function pointers which, when called, directly modify the scene in the game engine.

3.5 System Architecture

An initial system architecture design was completed with guidance and approval from the client, a diagram of which follows:

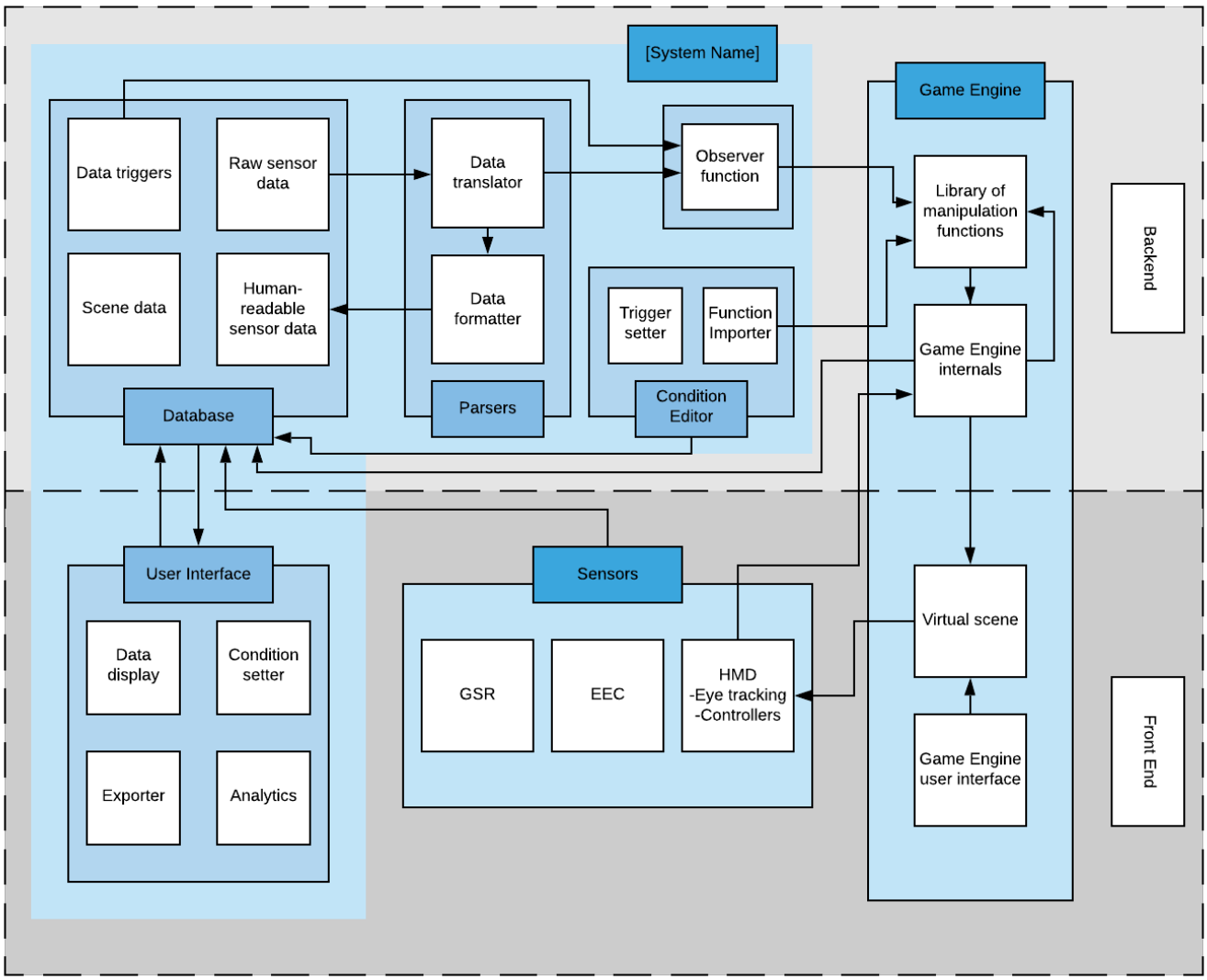

Fig. 4: System architecture diagram.

4 Code Samples

4.1 Dynamic Rain

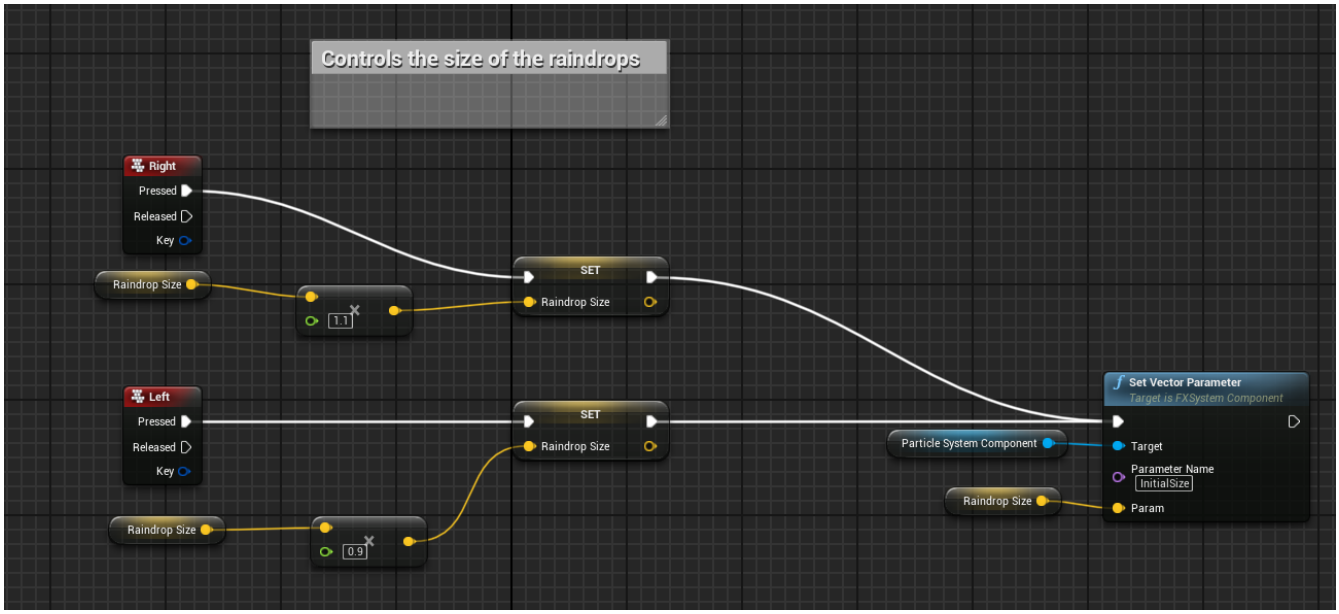

Fig. 5: An Unreal Engine 4 Blueprint which controls the size of raindrops in realtime.

Figure 5 shows how the Unreal Engine 4 rain particle effect can be modified in realtime. A variable called Raindrop Size is increased when the right arrow key is pressed, and decreased when the left arrow key is pressed. These events can be replaced by other events in the future. After the global variable Raindrop Size is set, each particle system needs to have the InitialSize parameter changed to match Raindrop Size.

5 RETROSPECTIVE

Name	Positives	Deltas	Actions
Kyle Hiebel	• Found existing code online which reduces workload for the team • Started writing assignments early • Group leader for the term, made sure all assignments got completed on time • Increased customer satisfaction by exhibiting enthusiasm for the project	• Improve communication with the customer • Stop overlapping with teammates' work and let them work independently • Keep the team oriented and working towards the same goal, reduce confusion	• Beginning of week slack messages indicating required work for that week • Plan work in a scrum agile environment such as Jira • Provide teammates who cannot attend class with lecture notes
Ayush Choudhury	• Completed documentation in a timely manner • Researched and found preexisting code that could be used to speed up production	• Better client communication • Some documentation needs more detail • Have more productive team-only meetings	• Give regular weekly updates to client about personal progress • Communicate with teammates about collaborating on the implementation
Ley Aldinger	• Informed the group about the history of the project and the overall goal • Worked with the client to design a system architecture	• Improve communication with the team, letting them know when I cannot complete an assignment	• Start working on assignments before the due date • Message on Slack to update team on my progress

Zunyue Qiu	• Read many research articles to understand the importance of time perception and why someone would want to measure time perception of VR subjects • Found additional research online about how to build a good API before starting to code	• Need to improve my communication skills during client and group meetings • Need to spend more time coding	• Send emails to the client to verify that my research matches with their intention • Start coding stories early which leaves time for debugging and testing