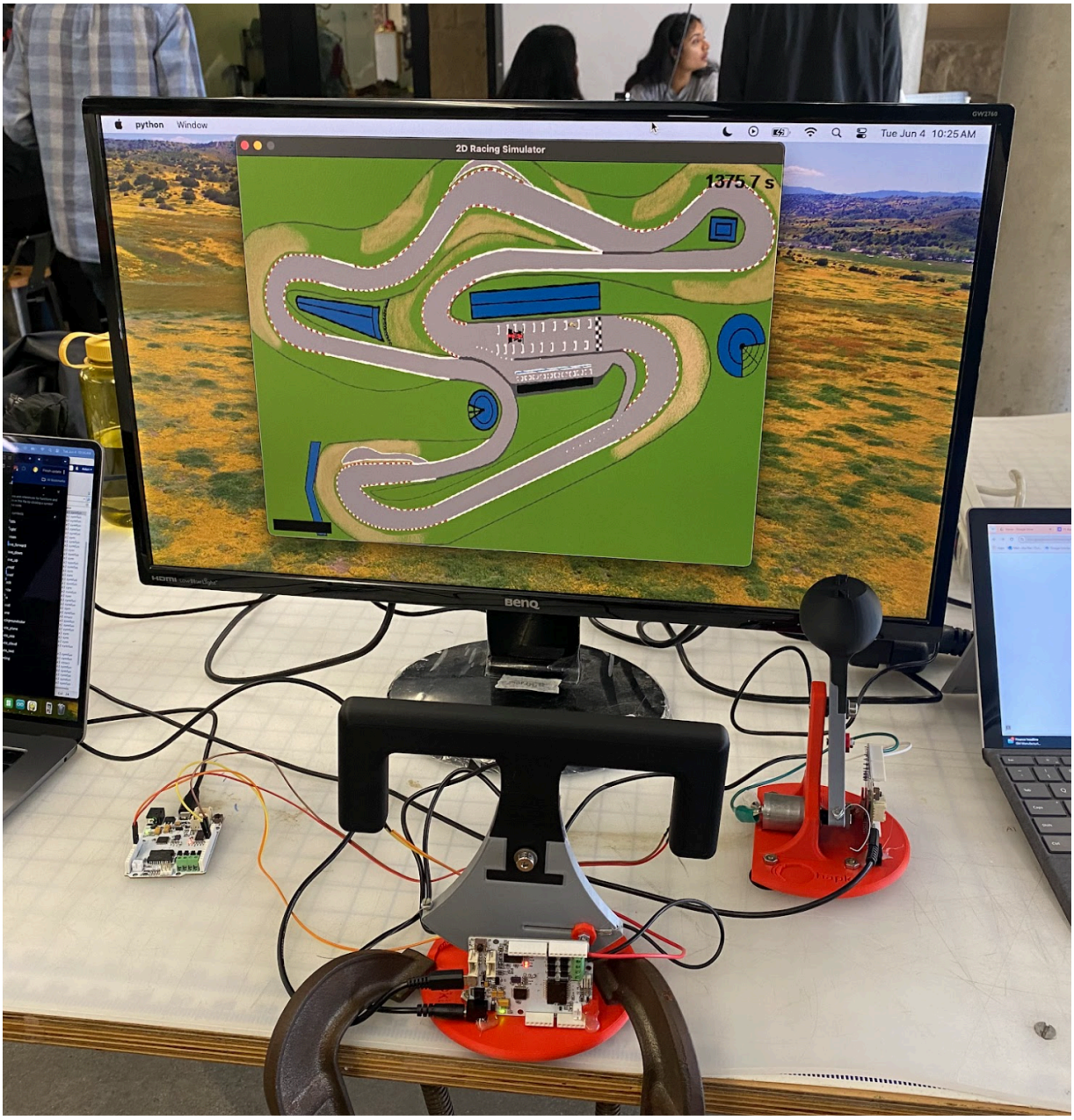
Project Report

Project Title: F1 Racing Simulator
Showcase Image:



Introduction

We have chosen to design and implement a haptic steering wheel accompanied by a simulated 2D racing game. The device aims to mimic realistic vehicle dynamics and provide corresponding feedback through the steering wheel. Our goal is to investigate the impact of realistic steering

dynamics and haptic feedback on driving simulation performance. This research could be pertinent to future drive-by-wire systems and their role in enhancing road safety.

Background

Nowadays, driving simulators combined with haptic feedback are widely used in games and driving lessons. However, creating simulators with proper feedback control remains challenging. To fully understand the methods and increase the realism of the device, the team has carefully studied several papers on related topics, including the nonlinear dynamics of a car model [1], self-motion sensations in driving simulators [2], and the development of a racing simulator game [3].

The first paper[1] covers a large amount of content regarding the dynamics of a vehicle. The primary focus on this thesis is on ABS and the necessary tradeoffs between vehicle stability and steerability during situations in which the system is activated. As this is an extremely lengthy document this summary will focus on the part most relevant to our project, which is the nonlinear bicycle model with slip vehicle dynamics. Per its namesake it assumes a 2 wheel system with both front and rear wheels represented by a single pair of wheels, 1 for the front and 1 for the rear. This model also assumes negligible lateral weight, roll, and that the road is flat. This vehicle model has the following state variables, the x and y position of the vehicle, the longitudinal and lateral velocities, along with the yaw and yaw rate. The inputs to the system are the steering angle and the longitudinal force. The slip angles of the two tires are given as the angle between the velocity vector and the tire angle. According to a tire model, lateral forces are then applied to the tires. The paper then suggests a linearized version of this model which is useful for stability analysis or speed. Overall this paper presents the equations necessary to simulate the nonlinear dynamics of a car model in 2D.

The second paper[2] presents a cost-effective approach to enhance the sensation of self-motion in driving simulators using force feedback and haptic motion. A standard gamepad (Microsoft Xbox 360) is connected to a 3DOF haptic device (Geomagic Touch) to simulate vehicle acceleration and turns. The force feedback applied to the user’s hands, proportional to the vehicle's acceleration, mimics the physical sensations of driving. Two force-feedback models were tested: the same direction model (aligns with acceleration) and the opposite direction model (opposes acceleration). Testing with 23 participants showed that haptic feedback significantly improved motion realism and user involvement, with preferences evenly split between the two models.

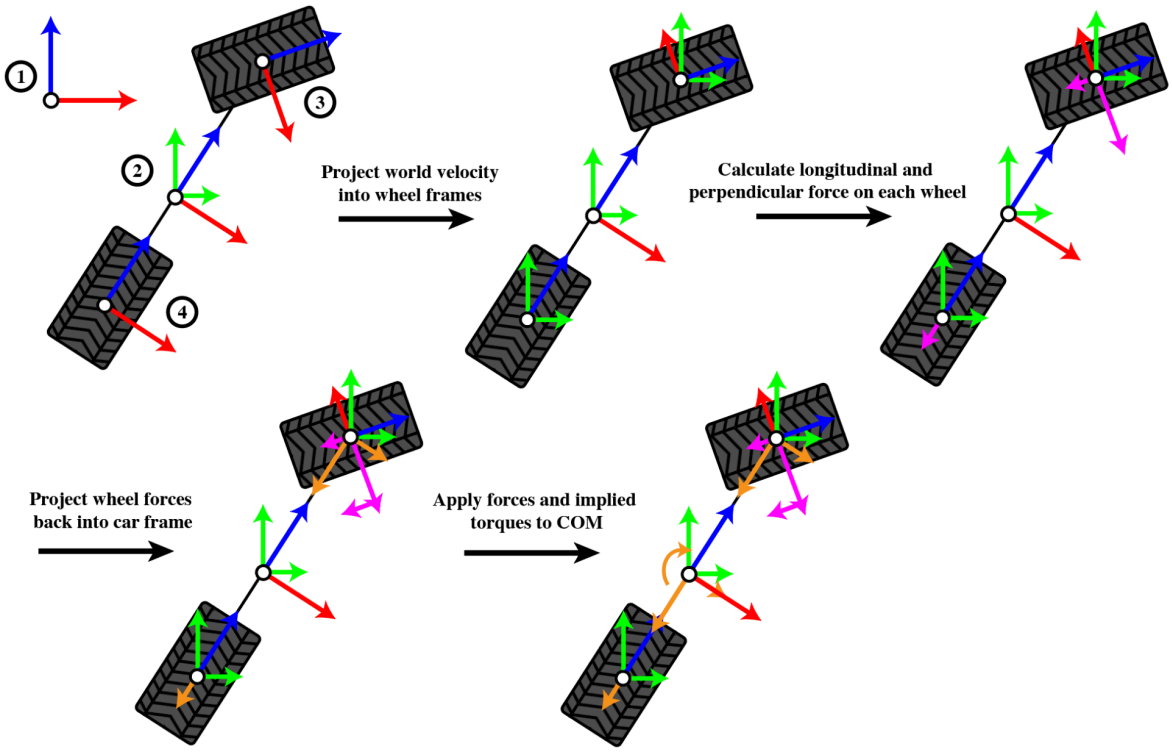
As for our project, by incorporating force feedback proportional to the car's acceleration and turning, we can significantly enhance the realism and immersion of our simulator. This can be achieved by integrating haptic feedback into the steering wheel, gear shift, and gas pedal.

The third paper [3] introduces *Racer*, a three-dimensional (3D) racing simulator game, providing a comprehensive insight into its design and implementation. It begins by detailing the software system's architecture, then explores various AI techniques applicable to the game's design, and concludes with a discussion on integrating these techniques into the software. The paper not only introduces the methods and tools utilized but also offers a step-by-step guide on creating the game. It delineates three main modules: the player-controlled car module, game-controlled car module, and environment module, elucidating their functionalities and interactions. For our project, this paper can serve as a roadmap for creating racing games, offering insights into utilizing existing resources and adopting a structured approach to development.

Methods

Dynamics

The dynamics are responsible for receiving the steering angle and throttle as input, then using this information to update the state of the virtual vehicle, and finally outputting a torque and vibration to the haptic wheel. We base our vehicle dynamics model on [1] which introduces a simplified 2D vehicle model. Consider the simplified 2 wheel bicycle model below.



We have four relevant frames which we need in order to properly calculate the dynamics. First we have the world frame, which represents the track. We also keep track of the velocities and positions of the COM of the car in the world frame. Next we have frame 2 which is the COM of the car as well as 3 and 4 which are the frames of the two wheels. Each time step we follow the procedure in the above figure in order to update our dynamics model. We will walk through this procedure step by step.

We first need to take our world velocity and express it in frame 2. This gives us the forward and perpendicular velocity of the car. Using this we express these velocities in frame 3 and 4 which gives us the forward and perpendicular velocities for each wheel. Here theta is the angle of the car while phi is the angle of the front wheels.

$$\mathbf{f}_{\text{world}} = \mathbf{R}(-\theta - \phi) \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{s}_{\text{world}} = \mathbf{R}(-\theta - \phi) \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$v_{\mathbf{f}} = \mathbf{f}_{\text{world}} \cdot \mathbf{v}, \quad v_{\mathbf{s}} = \mathbf{s}_{\text{world}} \cdot \mathbf{v}$$

The base of the car may also be rotating which would induce an additional side velocity in the wheels. We calculate it as follows and transform it in a similar manner as above.

$$v_{\text{ang}} = \omega \times \mathbf{r}_{\text{wheels}}$$

We use the perpendicular and forward velocities for each wheel to calculate the force experienced. For the perpendicular force we multiply the coefficient of friction by the slip speed. For the longitudinal force we first calculate the difference in forward speed with the spinning speed of the wheel. Here v_r represents the velocity of the rubber contact patch due to the rotational speed of the wheel.

$$f_{\text{force}} = -v_f + v_r, \quad p_{\text{force}} = -v_s \times 2.5$$

Next we apply a friction limit to simulate realistic drifting wheel slip. We define a scalar magnitude which the wheel forces are clipped to when exceeding. We adjust this friction limit dynamically depending on whether the car is on track or in the grass.

$$F = \sqrt{f_{\text{force}}^2 + p_{\text{force}}^2}$$

$$\begin{cases} f_{force}, p_{force} & |F| < F_{limit}, \\ f_{force} = f_{force} F^{-1} F_{limit}, & \text{if } |F| > F_{limit} \end{cases}$$

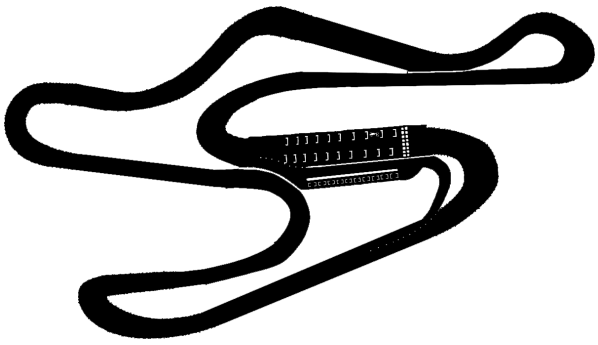
Finally given the net applied force on each wheel we sum to calculate the net force and torque on the car. This is then integrated using semi implicit euler integration to determine the position and velocity of the car as a function of time.

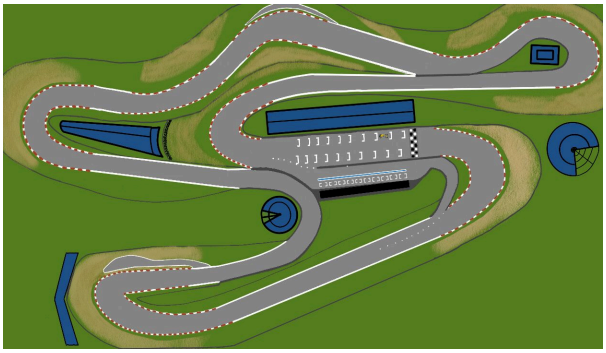
$$\tau_{\text{steering}} = -2 \cdot r_{wheels} \cdot \left(\arctan\left(\frac{v_y + \omega r_{wheels}}{v_x} \right) - \phi \right)$$

We implement a slightly simplified version of this in our code as we already calculate the perpendicular and forward velocities of the wheel in the car frame.

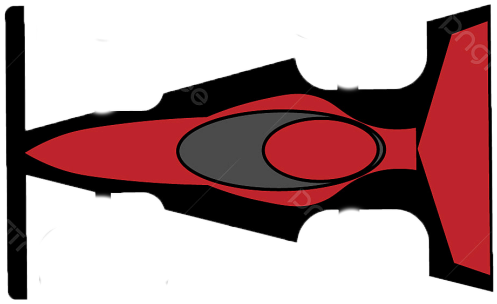
Simulated Environment

We utilize the pygame library in python in order to display graphics of the track and the car to the user. Use simply use a background image for the track, along with a mask which notates on track vs off track areas. When the car is off track a signal is sent to the haptic to trigger the vibration motors on the wheels.





We display the following car image on the track depending on its internal state. We draw each of the wheels individually depending on their steering angle to improve visualization for the user.



Communication Between Arduino and Python Code

We use three Hapkit boards (referred to as Arduinos later) in total for our setup and communication with Python code. The first Hapkit board is responsible for steering. It is directly attached by a steering wheel.The board receives the force value from the Python code and actuates the motor in the Hapkit to apply torque on the steering wheel. The force is calculated based on the friction on the wheel and the speed of the car in the Python code and the torque is calculated in the Hapkit board. The board then sends back the rotation angle to the Python code to control the direction of the car.

The second Hapkit board sends the xh value to Python for controlling the speed of the car without receiving any input from Python. It only controls the car to move forward and brake. The speed of the car will gradually increase as the value of xh increases, and the car will only brake at the most negative point.

The third Hapkit board receives commands from Python to initiate the vibration motors when the car is off track. Initially, we placed the vibration motor on the steering board, but it interfered with the magnet sensor, causing the torque feedback to be inaccurate. Therefore, we use this additional Hapkit board to control the vibration motors exclusively.

```python
data_queue1 = queue.LifoQueue()
data_queue2 = queue.LifoQueue()
def read_from_arduino(arduino, queue):
    while True:
        if arduino.in_waiting > 0:
            try:
                data = arduino.readline().decode('utf-8', errors='ignore').rstrip()
                if validate_data(data):
                    queue.put(data)
                    # print(f"Data received from Arduino: {data}")  # Print statement for logging
            except UnicodeDecodeError:
                continue

def validate_data(data):
    try:
        float(data)
        return True
    except ValueError:
        return False
```

Python code for reading and validating data from Arduino using a LIFO queue

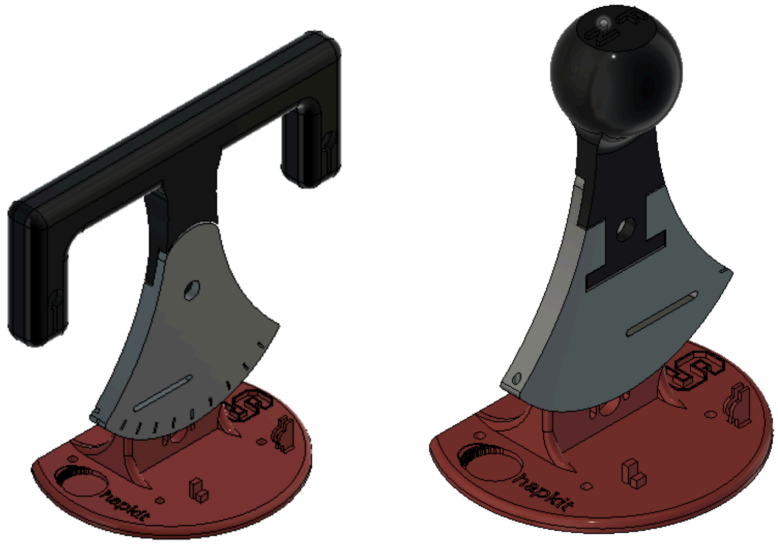
We use three ports for communication between the Arduino and Python, and specify the port number in the python code. The code employs a LIFO (Last In, First Out) queue to store incoming data from the Arduino boards, ensuring that only the most recent and relevant data is processed. The read_from_arduino function continuously reads data from the specified Arduino and validates it using the validate_data function, which checks if the data can be converted to a float. Valid data is then placed into the respective queue.

```python
while not data_queue1.empty():
    data1 = data_queue1.get()
    try:
        data1_float = float(data1)
        data_received1 = True
        continue
    except ValueError:
        pass
```

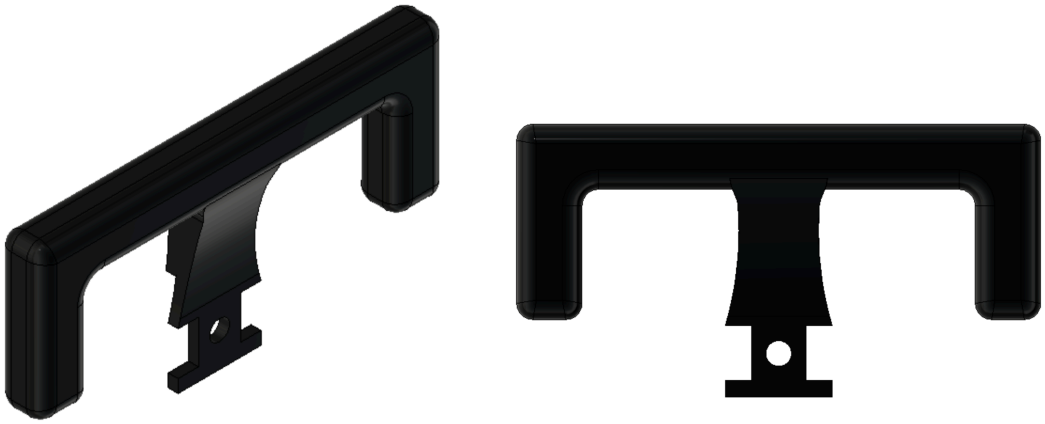
Python code for extracting and validating data from data_queue1, ensuring only valid floating-point numbers are processed

For each queue, the code executes a loop to continuously check for new data. The loop begins by extracting the most recent data from the queue using the get method. It then attempts to convert this data into a float. If the conversion is successful, a flag is set to True, indicating valid data has been received. If the conversion fails (resulting in a ValueError), the invalid data is ignored, and the loop continues to the next iteration.

Design



In building the driving simulator, it was necessary to create an upgraded handle for the haptic system to enhance the realism. It was obvious that a steering wheel was essential. Initially, the steering wheel was designed to resemble that of an everyday passenger vehicle—circular in shape. However, this design was quickly modified to a rectangular shape commonly seen in race cars. The rectangular design was 8 inches wide, with handles measuring 3 inches in length. The cross-sectional area of the handles is approximately 1 square inch, making them comfortably thin for any hand size.



The back of the steering wheel features two press-fit slots to accommodate vibration motors. This design allows the vibrations to be felt throughout the steering wheel, as they permeate through the 3D print. The tight fit of the motors prevents any rattling sounds from loose components.

In addition to the steering wheel, our team needed an improved design for the hand pedal. We modified the standard handle to create a sphere with a diameter of approximately 2 inches. The top surface was flattened, and we embossed the letters “F” for forward and “N” for neutral on the top. This design was both ergonomic and effective in displaying the purpose and functionality of the hand pedal.

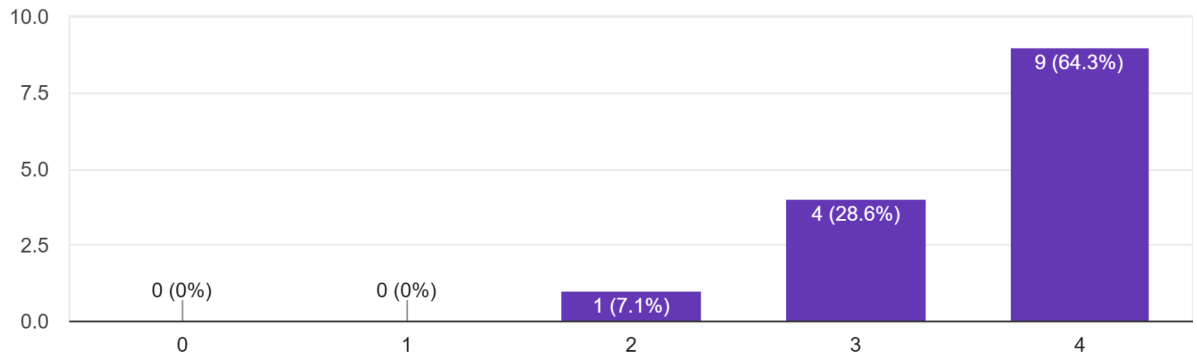
Results

On open house day, the driving system was successfully set up for users to experience the racing simulator. Users could control the car's direction and speed using the steering wheel and hand shaft. Most users felt the vibration when the car went off track and experienced a large torque applied to their hands at high speeds. However, there were instances where the steering wheel misaligned with the Hapkit output when moved too quickly or aggressively, causing some users to be unable to finish the lap. Overall, most users enjoyed their racing experience and provided proper feedback of the realism on the vibration, torque and speed control.

To gather feedback, we conducted a survey with the participants. Fourteen individuals participated, providing valuable insights into their experience with the setup.

How would you rate the realism of the vibration within the context of the simulation?

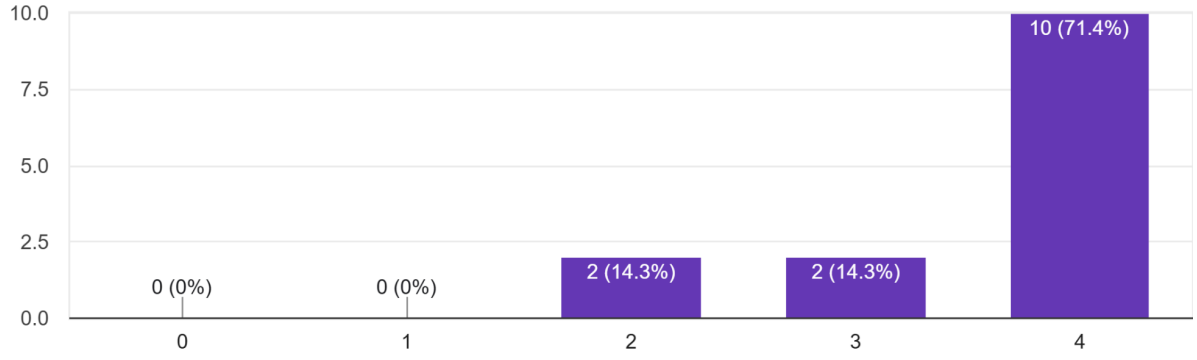
14 responses



The majority of participants (64.3%) rated the vibration realism highly at 4. This indicates a strong overall positive reception regarding the simulation's vibration realism, with nearly all participants giving ratings of 3 or above, showcasing its effective implementation.

How would you rate the realism of the steering wheel torque within the context of the simulation?

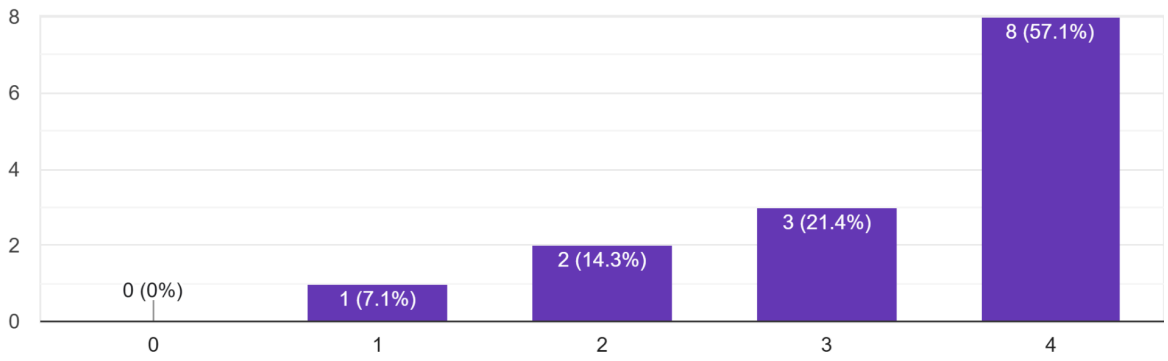
14 responses



A significant proportion of participants (71.4%) rated the steering wheel torque realism at 4, with no respondents rating it below 2. This suggests that the simulation effectively mimics the real-life steering wheel torque, contributing positively to the immersive experience.

How would you rate the realism of the speed control within the context of the simulation?

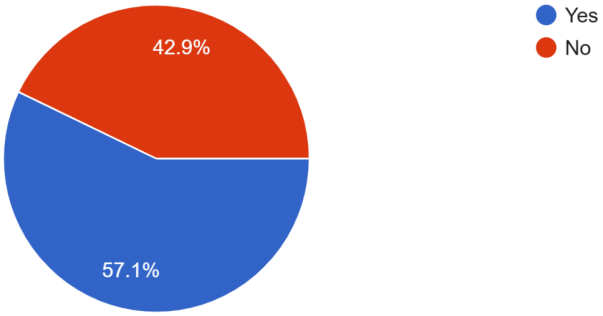
14 responses



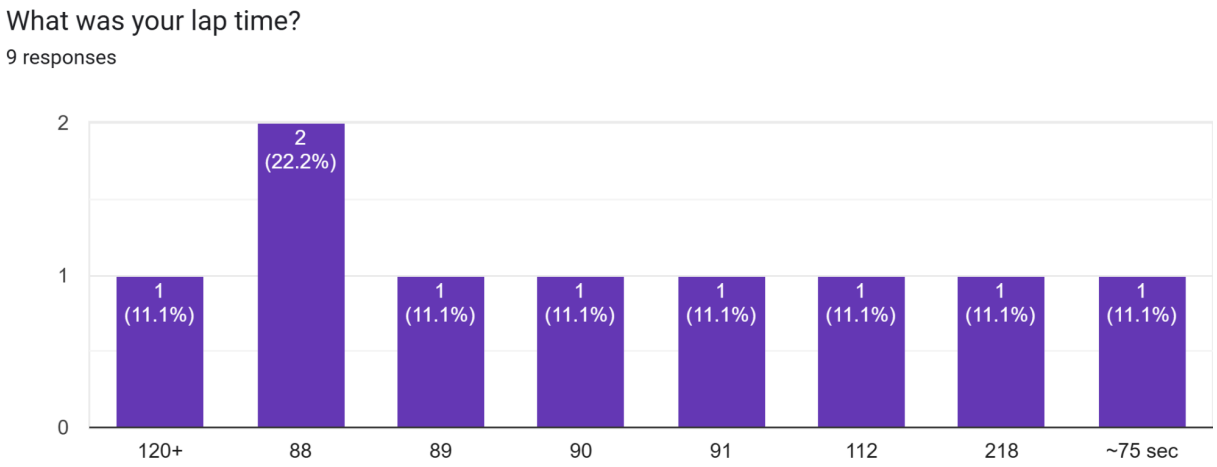
Over half of the respondents (57.1%) rated the speed control at 4, with the majority rating it at least 3 or higher. This indicates that the speed control aspect of the simulation was largely perceived as realistic by participants.

Were you able to finish a lap?

14 responses



A majority of respondents (57.1%) were able to finish a lap, while a significant minority (42.9%) were not. This highlights a potential area for improvement in the simulation's usability or difficulty level to ensure more participants can complete a lap.



The distribution of lap times varies significantly, with times ranging from approximately 75 seconds to over 218 seconds. The most common lap time recorded was 88 seconds (22.2%), but the small sample size (9 respondents) and wide range of times suggest a high variability in participant performance. This could indicate differences in participant skill levels or the complexity of the simulation.

The survey results indicate a generally positive reception towards the simulation's realism, with most participants rating the vibration, steering wheel torque, and speed control highly. However, a notable portion of users struggled to finish a lap, suggesting room for improvement in usability. Lap times varied widely, indicating differences in participant skill levels or simulation complexity. Overall, while the simulation is effective in providing a realistic experience, enhancements are needed to ensure a more consistent and accessible user experience.

Future Work

Future work will focus on several enhancements to improve the simulation experience. First, we plan to add bumps to the gas pedal, simulating the feel of different gears (e.g., first and second gear) to provide varying speeds. Upgrading the hardware components to increase reliability is also a priority, ensuring the system can withstand aggressive use without misalignment or delays. This would first take the form of an upgraded encoder. The hull effect encoder on the board was unreliable in the context of a racing simulator which had a negative impact on the realism of the steering wheel. Additionally, expanding the range of tracks and racing car options would add variety and excitement for users. These improvements aim to create a more immersive and realistic simulation experience.

Acknowledgements

We'd like to acknowledge the CAs—Yiyang, Danny, and Ankitha—for their support during the quarter and in developing this project. In particular, we extend our thanks to Allison for organizing a wonderful class that provided us with the ability and opportunity to create a fun project.

Files

https://github.com/swannaiden/haptics_final_project

https://github.com/shaneblinkman/me327_final_project

References

[1] Taheri Saied, An investigation and design of slip control braking systems integrated with four-wheel steering, PhD Thesis Clemson University, 1990. https://www.proquest.com/docview/303866547?pq-origsite=gscholar&fromopenview=true&sourcetype=Dissertations%20&%20Theses

[2] G. Bouyer, A. Chellali and A. Lécuyer, "Inducing self-motion sensations in driving simulators using force-feedback and haptic motion," 2017 IEEE Virtual Reality (VR), Los Angeles, CA, USA, 2017, pp. 84-90, doi: 10.1109/VR.2017.7892234. https://ieeexplore.ieee.org/abstract/document/7892234

[3] Chan, Marvin T., Christine W. Chan, and Craig Gelowitz. "Development of a car racing simulator game using artificial intelligence techniques." *International Journal of Computer Games Technology* 2015 (2016): 12-12. https://www.hindawi.com/journals/ijcgt/2015/839721/