CS3483

MULTIMODAL INTERFACE DESIGN

FINAL REPORT

Group Members:

SHYLA KUMAR Rohit (54581876)

TANDON Rishabh (54106672)

KABANI Sameer (54719396)

SRIDHAR Prashant (54565839)

CHOUDHURY Archit (54562063)

User Community

Video games have always encouraged users to try new ways to achieve the same objective. As the industry grew more popular, various interface systems were designed so that the user's movements in the game feels more natural and fluid. With the introduction of smartphones, video games could explore new avenues for innovation with the help of the accelerometer and gyroscope. The mobile racing game genre of video games was where such innovation could thrive as the user could move his/her device in a way that mimics natural actions (the movement of the device is tracked by the gyroscope and accelerometer). This is how we were able to find our niche as these users would be accustomed to trying out new interactions techniques that mimic natural movements.

Our targeted consumer base would comprise of individuals that are targeted to play first or third person racing games on their mobile phones. From research, this covers individuals between the ages of 6-60 if we look at both the extremes, but realistically this falls somewhere between 12-40.

Our project requires users to have good hand-eye coordination skills and dexterity. For this reason, we would logically exclude people over the age of 40 from our target consumer base because studies show that the motor functions required for such activities decrease in effectiveness over the age of 40.

Furthermore, our gestures are very similar to the gestures one would use when playing a motion-sensing racing video game on their smartphone. We see that the target consumer base for mobile phone racing games falls between the ages of 12-40 with the ages of 16-28 being the most active users.

So our strategy to choose the final consumer target group would be to cater to mobile racing game users between the ages of 12-40 with special efforts being made to focus on marketing to ages between 16-28 as they will form the bulk of our consumer base. People in this age group are influenced by motorsports such as Formula 1 and Nascar so advertising our project as a fun and realistic way to experience these sports can help drive demand for our project.

Considering how racing video games are quite popular around the world, specifically targeting a particular gender, country or region in the world would be counterproductive. This is why we would use neutral advertising themes focussed on Motorsports and Gaming (such as Formula 1) to attract a global audience.

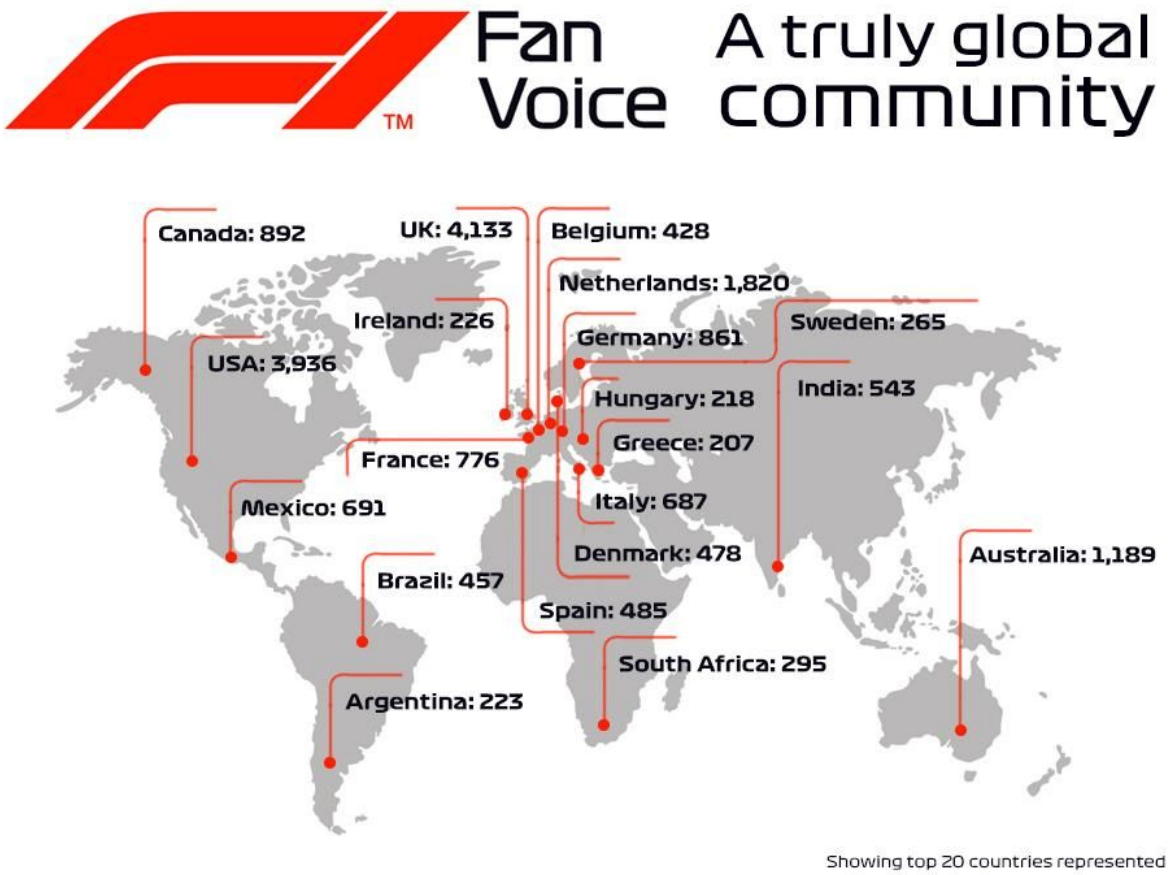


Image showing the number of dedicated Formula 1 groups in the top 20 countries that shows our marketing efforts can be globalized

Screens and the relationship between them

Our main design involves five screens and the interaction between them allows the user to play a game using gestures. Our application also has a thread that is always listening for commands. Some of those commands lead the user to one of the screens, corresponding to their selection. The screens are as follows:

Welcome Screen

The Welcome screen is the first screen the user sees when they first run the application. The welcome screen first asks the users to accept terms and agreements to use the program. The user can click the ‘Get started’ button to proceed to the menu screen. Here the user will be able to select from the multiple options on how to proceed.

While proceeding forward the program asks the user if they have ever used an application with gesture control. If the user is familiar with this technology, the program proceeds. Otherwise, the program recommends videos and reading documents for the user to view before using the application so they have some idea of how the technology works.

Menu Screen

The Menu screen is the second screen of the application. The user gets four choices in the form of buttons to decide how to proceed with the application. The buttons include:

- ‘Start Application’ button. This button allows the user to start using the main application. If the user is using the application for the first time during the day, the button will first lead the user to the calibration screen so that the user can set their position perfectly. If it isn't their first time, pressing the button will automatically lead the user the main screen so that they can play the game.
- ‘Exit’ button. Pressing this button will quit the application. The program will shut down when this happens. It will also inform the user that it is shutting down in the form of a voice message.
- ‘Help’ button. Pressing this button will lead to the help screen. The existence of this button is especially important as the help screen tells the users how to use the application to play the game. Users who are using the application for the first time/after a long period and need a quick tutorial on acceptable gestures and commands will find this button to be quite useful.
- The ‘Calibrate settings’ button. This button will lead the user directly to the calibration screen if the user wants to calibrate their position and make sure it is accurate.

Calibration Screen

The Calibration screen will load up a window with the game running along with an overlay of the video being captured by the webcam. This system works very similarly to how Kinect sensor requires the user to make sure they are standing in the correct position so that they can make maximum use of the kinect to play games. Similarly, the calibration screen shows a rectangle on the window and guides the user to make sure they always have their hands inside the rectangle. Doing that will allow OpenCV to maintain focus on the hands and recognize the various gestures the user is performing and respond accordingly. Once that is done, the interface asks the user to perform simple tasks such as accelerate or break in order to calibrate and at the same time introduce the gestures to the user. There is also a voice level check which asks the user to speak some sample sentences. The program does this to make sure the user is loud enough to be heard properly so that later when the user says a certain command, the program can hear and understand it, and respond accordingly. After all of this is done, the user can simply continue racing on the sample track or exit to the main screen so that they can have an actual race..

Main Screen

The main screen is divided into two parts. They are:

- The game screen. This part of the screen is displaying the game itself. The user can see their car in the game on the race track and depending on what part of the race is coming up next(straight path, turn, jump etc) and decide which gesture they would like to perform in response. The program responds to these gestures and controls the car accordingly.

- The info screen. This is a smaller half of the main screen in comparison to the game screen. This displays the live video of the user with a green rectangle being marked on screen to show where the hands are supposed to be. The user can use this to keep checking if their hand is in the correct position always and make adjustments. A statistics side also shows the speed of the car and other key stats that the user might be interested in. The sign next to the name of the user (sign next name silentsword) is shown when the program recognizes any voice commands. The info screen helps the user always be in the right position to play the game.

Help Screen

The help screen contains information regarding what gestures are available and instructions regarding how the user can execute it. These gestures are the most basic way the application allows the user to communicate with the game. The gestures shown in the help screen include:

- Gesture to accelerate forward. This gesture requires the user to hold up both their fists next to each other with no angle between them. The application will use the change in distance between the users fists to decide whether to speed up or slow down the car

- Gestures to turn right/left. These gestures require the user to hold up both of their fists next to each other in such a manner as to have an angle between them. The value of this angle decides whether the car in the game turns left or right.

- Gesture to brake/reverse. This gesture requires the user to hold up both of their fists next to each with no space between them. When this position is detected the car will be made to brake. If the user still continues with the same gesture, the car will then start to move in reverse.

The help screen also contains information regarding the multiple voice commands the user can use to interact with the application. The voice commands provide additional functionality to the application and come in handy while the user is playing the game. The voice commands that the user can execute are:

- 'Reverse'. Executing this command will run the program in reverse mode. In this mode every gesture will execute completely opposite of what they were originally meant to do. To quote a few examples, the gesture to go left will turn the car right and vice versa.

- 'Forward'. Executing this command will quit the reverse mode (if the program was originally in this mode) and every gesture will once again start executing what their original task was

- 'Pause'. Executing this command will pause the game and the application as well. It will come in very handy if the user has to suddenly attend to an alternative matter. However, even in this case the speech recognition program is still working and listening for commands to execute.

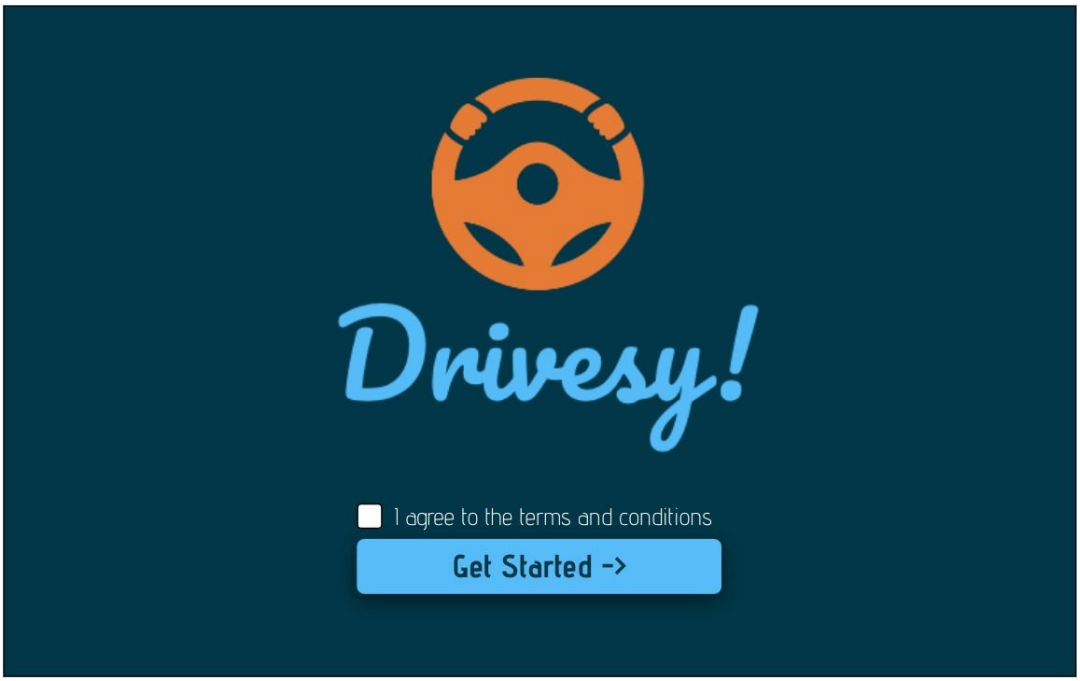
- 'Resume'. Executing this command will once again resume the application and the game (which were in pause mode)

- 'Stop'.Executing this command will instruct the program to shut down. The program will shut down when it hears this command. It will also inform the user that it is shutting down in the form of a voice message.

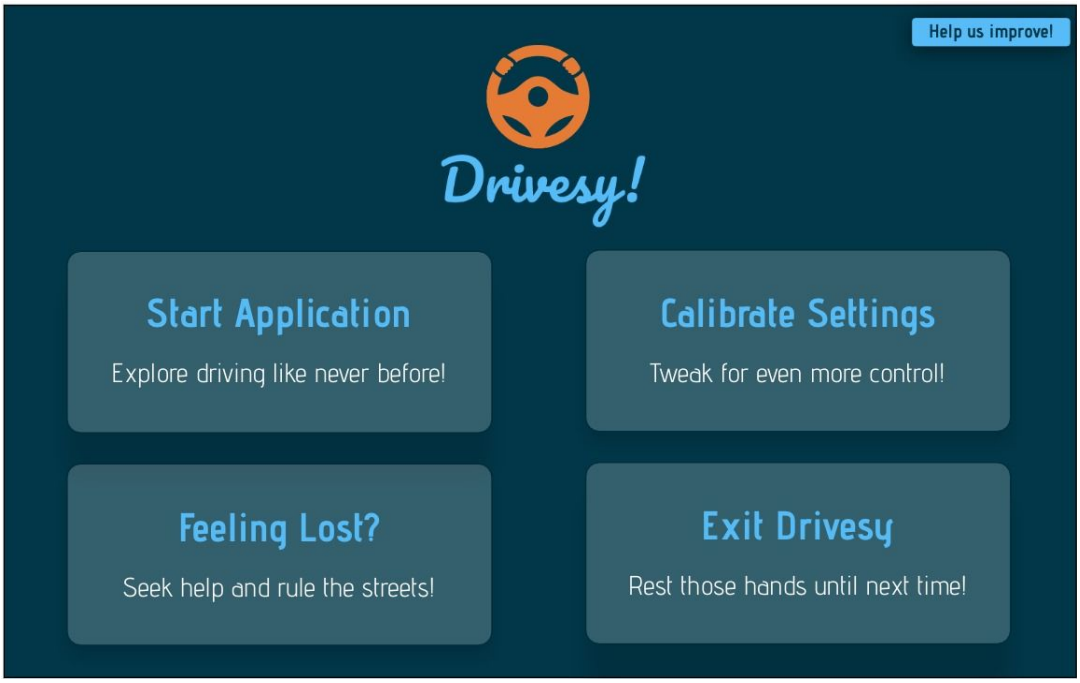
- 'Help'. Executing this command will lead the user to the help screen where they can take a look at the commands and gestures available to them. While this is happening, both the game and the application go into pause mode. This comes in very handy if the user wants to look up specific gestures or commands while playing a game.

ScreenShots

Welcome Screen

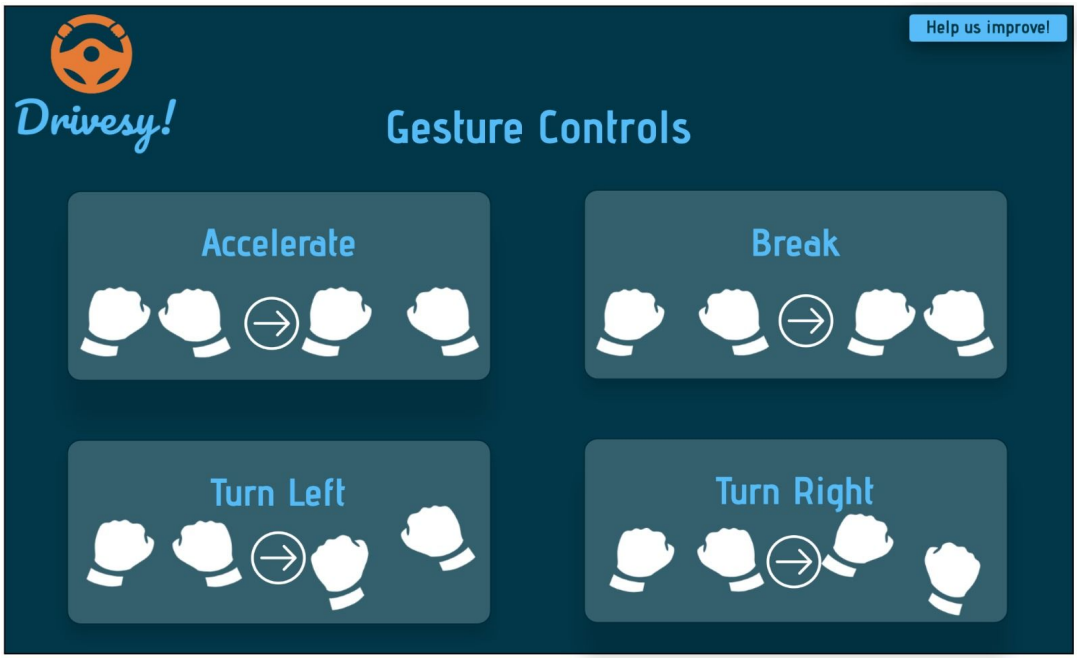


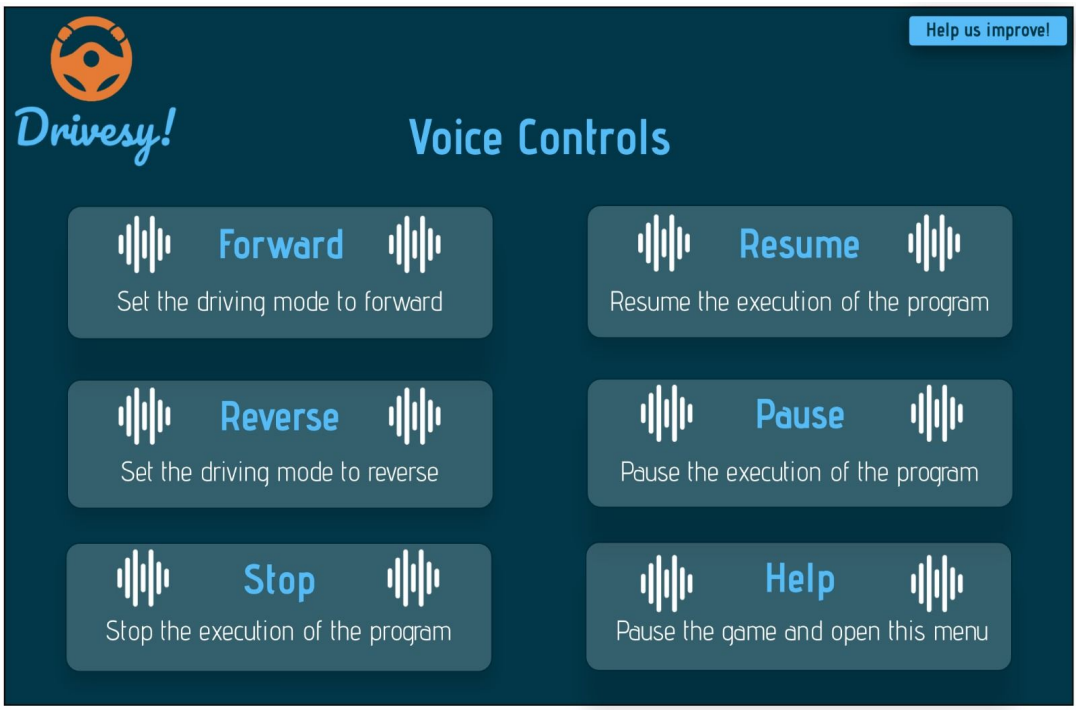
Menu Screen



Main Screen

Help Screen 1 & 2





Calibration Screen

Principles and Guidelines used

When we used the GOMS (Goals, Operators, Methods & Selection Rules) to initially formulate our interface design. We used this model because it allows us to effectively predict our user's actions with a high degree of accuracy without actually having our users participate during the design phase of our program. Additionally, our applications require users to perform a set of repetitive gestures and tasks so there is a 'model answer (gesture)' to complete each task. Using GOMS allows us to incorporate this into our application effectively.

Goals

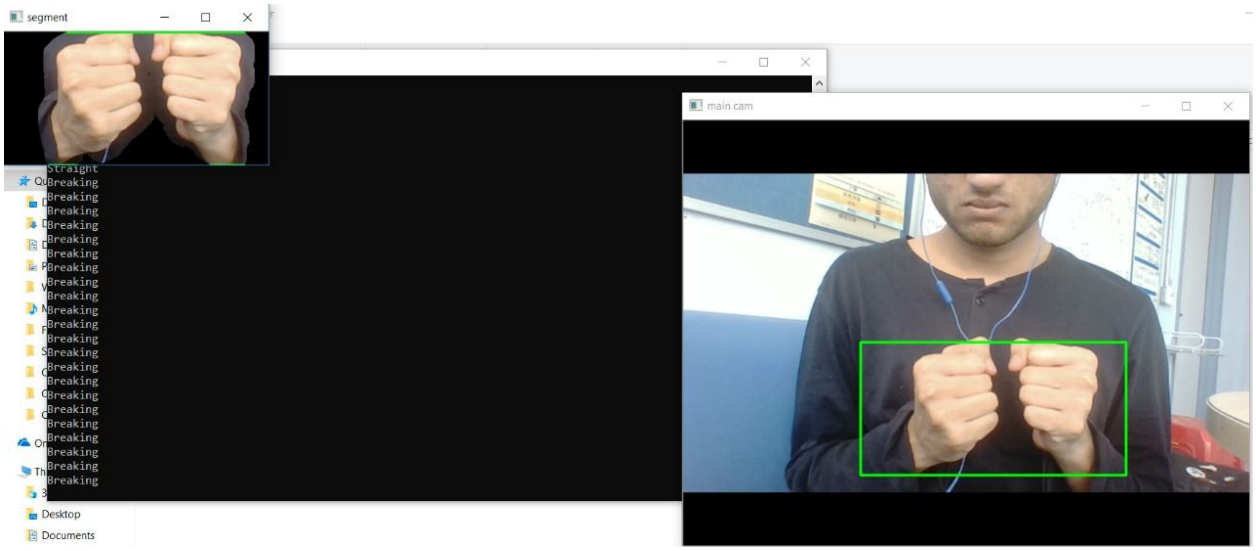
The goal of the user is relatively straightforward - to play the racing video game in a manner that seems natural and easy to learn. We allow users to achieve this goal with the help of gestures. When we think of racing video games, there are five main goals that the users want to achieve with the vehicle - moving forward, moving left, moving right, braking and reversing.

Operators and Methods

Now that we have defined the four basic goals that users want to achieve, we shift our focus towards the physical actions that need to be taken by users to achieve these goals and how do they learn these physical actions to achieve their goals. We will address the operators and methods for each specific goal below: -

1. Moving the vehicle forward: -
    a. Method A (Gesture Recognition):
        i. The user needs to remember which gesture is used to accelerate the vehicle
        ii. The user needs to perform the selected gestures in order to accelerate the vehicle.
        iii. The user needs to control the level of acceleration achieved by the vehicle using gestures
        iv. The user accomplishes the goal of moving the vehicle forward
    b. Operators used (For Method A):
        i. The user can accelerate the car by spreading his/her fists apart.

ii. The user can control the degree of acceleration by controlling the distance of separation between his/her fists. The wider apart the fists are, the faster the car will accelerate and vice versa. There is a maximum distance of separation between the two fists that achieves the maximum possible acceleration. We do this so that the user's fists still remain within the frame the video recording device.



c. Method B (Speech Recognition):

i. The user needs to remember the voice commands used to access various features of the interface.

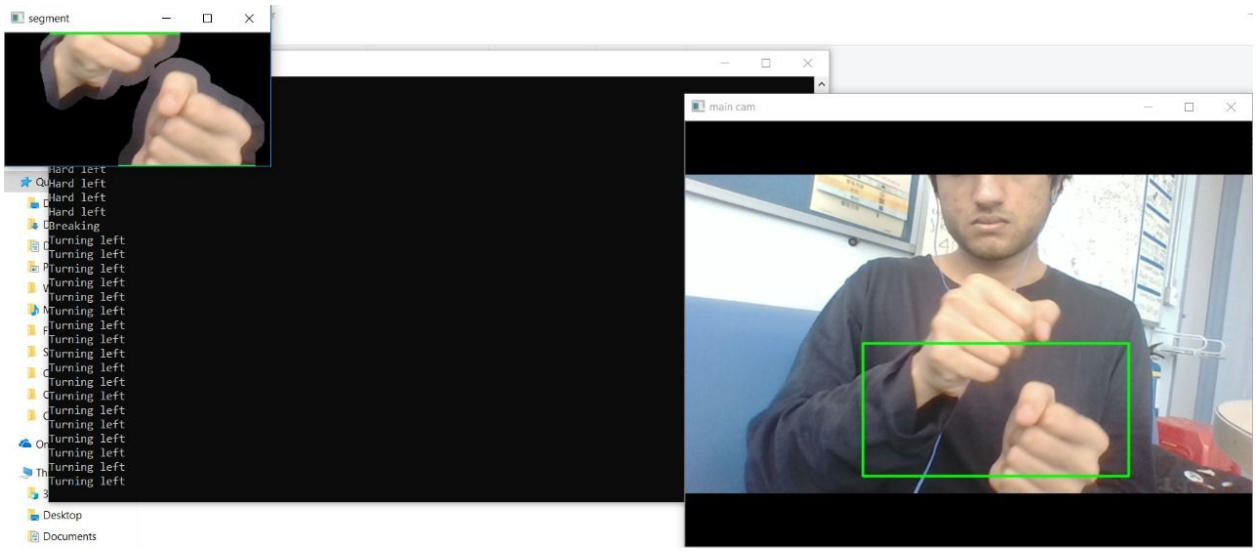
ii. The user needs to clearly recite the voice command that is used to trigger the vehicle in the game to move forward if the vehicle was in reverse mode

iii. The user accomplishes the goal of moving the vehicle forward.

d. Operators used (For Method B):

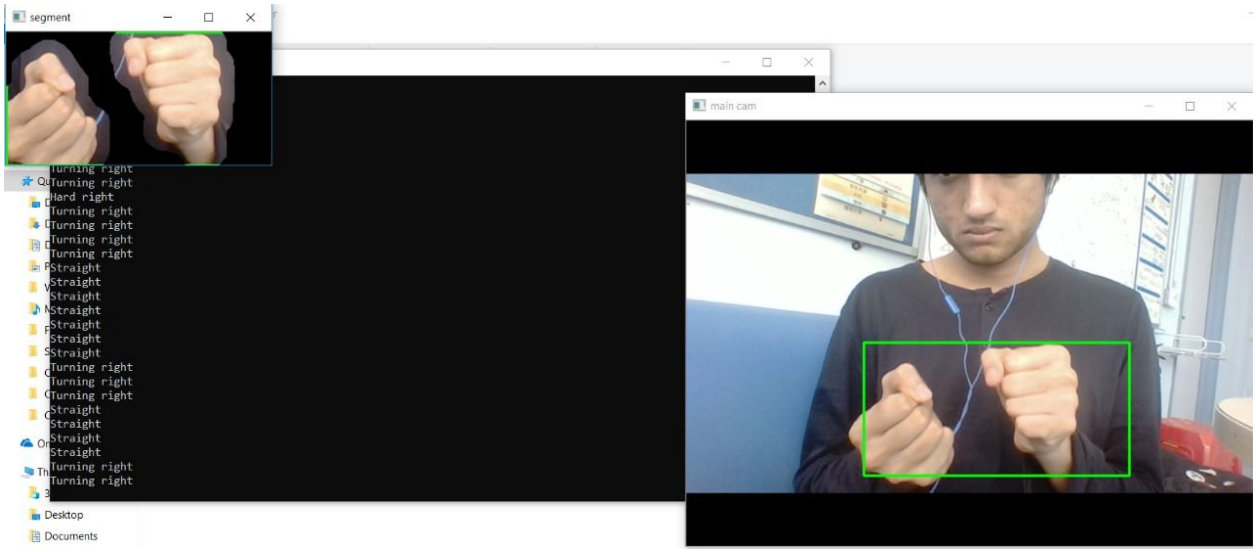
i. The user can cause the vehicle to move forward by reciting the keyword 'Forward'. The program will recognize that the user wants to move the vehicle forward and thus the vehicle in the game will begin to move forward at the maximum possible rate. The user is unable to control the degree of acceleration in this mode.

2. Moving the vehicle left:-
    a. Method A (Gesture Recognition):
        i. The user needs to remember which gesture is used to move the vehicle towards the left-hand side
        ii. The user needs to perform the selected gestures in order to move the vehicle to the left.
        iii. The user needs to control the level of cornering achieved by the vehicle using gestures
        iv. The user accomplishes the goal of moving the vehicle towards the left
    b. Operators used (For Method A):
        i. The user can move the vehicle by moving both fists to the left. This action mimics the actions the users will make in real life if they were to move a steering wheel in a vehicle towards the left.
        ii. The user can control the degree of cornering by controlling the angle between the two fists. The sharper the angle between the two fists, the sharper is the degree of cornering and vice versa. There is a maximum angle of separation between the two fists that achieve the sharpest cornering angle. We do this so that the user's fists still remain within the frame the video recording device.



3. Moving the vehicle right:-
    a. Method A (Gesture Recognition):
        i. The user needs to remember which gesture is used to move the vehicle towards the right-hand side
        ii. The user needs to perform the selected gestures in order to move the vehicle to the right.
        iii. The user needs to control the level of cornering achieved by the vehicle using gestures
        iv. The user accomplishes the goal of moving the vehicle towards the right
    b. Operators used (For Method A):
        i. The user can move the vehicle by moving both fists to the right. This action mimics the actions the users will make in real life if they were to move a steering wheel in a vehicle towards the right.
        ii. The user can control the degree of cornering by controlling the angle between the two fists. The sharper the angle between the two fists, the sharper is the degree of cornering and vice versa. There is a maximum angle of separation between the two fists that achieve the sharpest

cornering angle. We do this so that the user's fists still remain within the frame the video recording device.



4. Moving the vehicle backward or braking:-

    a. Method A (Gesture Recognition):

        i. The user needs to remember that the same gesture is used to brake and move the vehicle backward.

        ii. The user needs to remember which gesture is used is used to brake and move the vehicle backward.

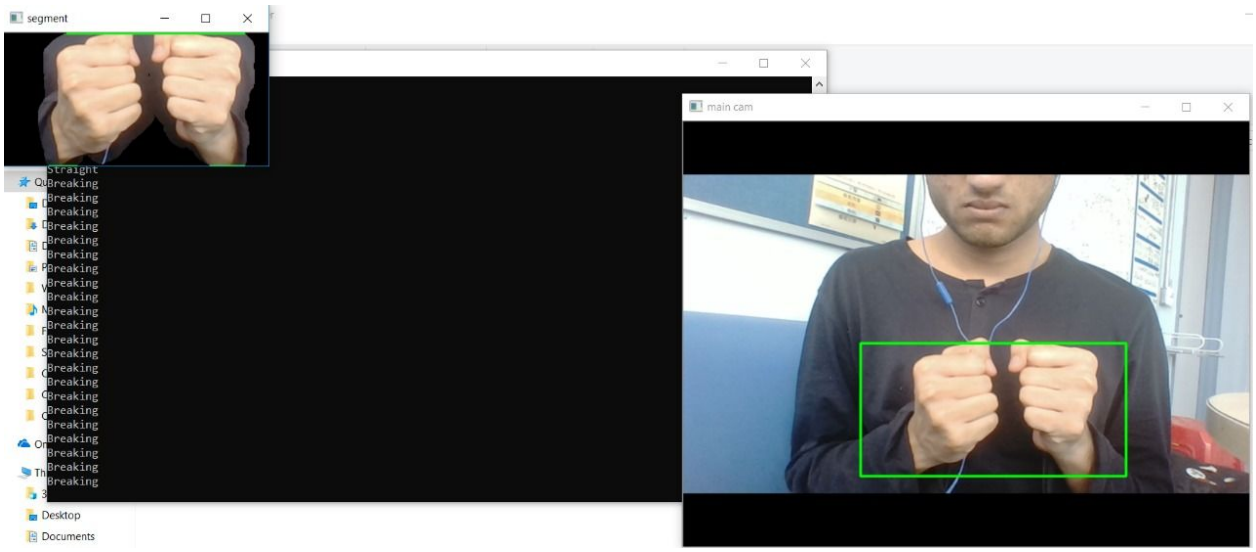
The user needs to perform the selected gestures in order to brake and move the vehicle backward.

        iii. The user accomplishes the goal of braking or moving the vehicle backward.

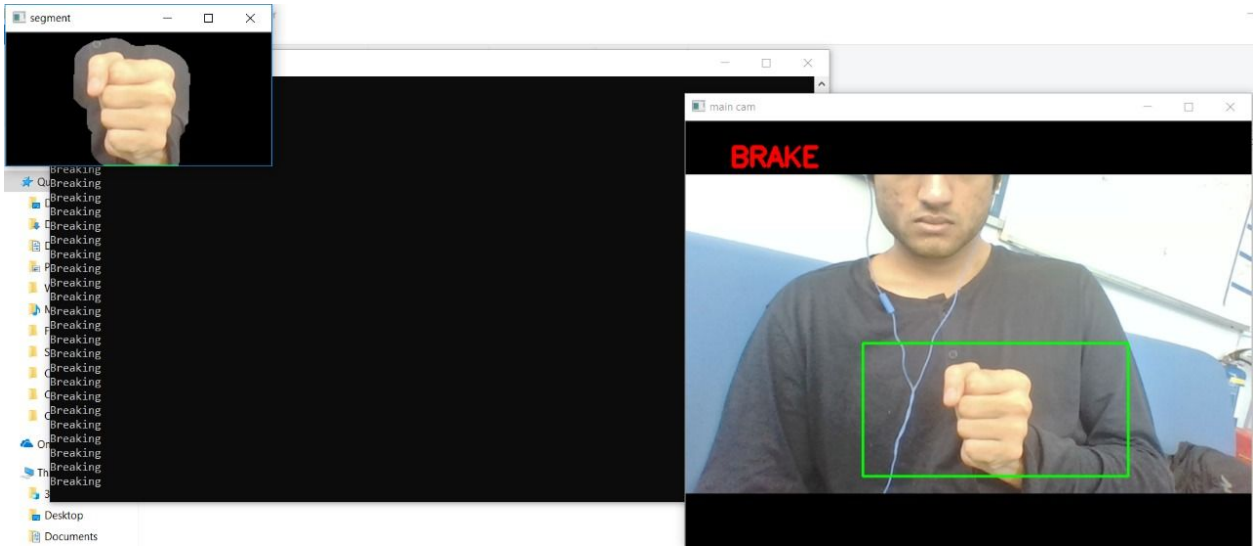
    b. Operators used (For Method A):

        i. The user can perform the brake function by bringing his/her fists closer together. By reducing the distance of separation between the two fists, the user will cause the vehicle to slow down. Once the vehicle comes to a complete stop and if the distance between the two fists is still minimal (or if

the distance is 0) then the vehicle will start moving backward at the maximum possible rate.



ii. The action of braking or reversing the vehicle can also be performed with one hand. As the user's fists move closer together, the number of contours in the scene reduces to 1. So this is equivalent to having only one hand up. Thus users have two possible ways to perform these tasks.



Selection Rules

The user can decide on which gesture to used based on the action that he/she wishes the vehicle to perform in the game.

Alternative Modal Techniques used

So in our project we have implemented interactions that conventionally aren't unique but are unique when it comes to our use case which is using them for playing racing based video games. These interactions were Gesture Recognition and Speech Recognition respectively. We use simple and intuitive gestures to let the end-user experience Racing based video games in a rather peculiar and fun way.

Here's a brief description regarding how we achieved these interactions:

Gesture Recognition

Using OpenCV, we first develop Contours of the fists within a particular Region of Interest. The reason we do this is to make sure that the computer only focuses on what the fists are doing and nothing else. After developing these contours, we then calculate the slope between the fists. This slope actually helps the computer identify whether the vehicle in the game is turning to particular direction. If the slope calculated between the fists were positive, the vehicle in the game turns left and if negative it turns right. To achieve this, we first obtain the moments (cv.moments) of both the contours detected from the region of interest(left hand and right hand) and using these moments we calculate the slope using the widely-known formula:

$$slope = tan\left(\frac{y2-y1}{x2-x1}\right)$$

Using this value, we will then know whether the vehicle turns left or right.

If the slope calculated was positive, then as previously mentioned the vehicle moves to the left. We simple map the keys that are normally used in games like these to move the vehicle left to the gesture. However, in this mode we have added a subtle feature. When we calculate the slope between the fists, we also calculate the angle between the captured contours of the fists. While the vehicle is turning left, if the angle between the fists while performing this gesture is greater than or equal to 65 degrees, then the vehicle will perform a 'sharp' or a 'hard' left. Similarly, the vehicle would take a 'sharp' or 'hard' right if the angle is greater than or equal to 65 degrees while turning right.

Speech Recognition

Another interesting mode we added when it comes to interacting with our interface is, making use voice to start the application, stop the application and also help users learn how the interaction works by simply having them say "help".

To offer this feature we implemented the Google Speech Recognition API, which records and recognises the voice in real-time and stores the sequence of words said to it in a string. We then split this string and iterate through it to see whether a particular keyword we wanted was said. In our case the keywords were “resume”, “stop”, “pause”, “help”, “reverse” and “forward”.

If “resume” was said, our interface will takeover the game and the user can then stop using their keyboard and start using their gestures to control the vehicle.

If “stop” was said, our interface will shut down and the user can continue using their keyboard to control the vehicle in the game.

If “help” was said, we open another window along with the game to show the user what are the gestures they need to use in-order to interact with the game.

If “reverse” was said during gameplay, our interface will switch to the reverse mode through which the user can now reverse their car and perform subsequent actions without any hassle whatsoever.

Finally during the “reverse” mode, if the user says “forward” our interface will revert back to its initial set of gestures and user can control or interact with the vehicle in the game as usual.

Future Extensions

A possible extension of our current design could be to use video games to focus on rehabilitating people with disabilities (such as veterans, people who suffered from accidents or people born with deformities). With the advancement in technology, many people with disabilities are able to procure prosthetic limbs but take years to physical therapy for their bodies to get acclimated to it.

Our program makes use of some elementary motor functions and thus with enough practice and repetition, the users should become comfortable with those movements. Our program can offer a cheap alternative to physical therapy while also entertaining the user at the same time. They can also use our application in any environment so they can save on transportation and accommodation costs as well. We can include training manuals in our application designed specifically for users with different disabilities so that they can learn in a more intuitive and easy fashion. These training manuals can focus on some basic movements at first and then move on to some of the more complex movements that require more control and synergy. We can expand from video games to more scenario based games where we place users in everyday scenarios that prompt them to use their limbs (such as picking up something of the top shelf of the kitchen, waving to a friend or playing ping pong). Virtual Reality can also be incorporated with the help of Oculus and Virtuix Omni so as to completely immerse the user in this artificial learning environment.

By focusing on rehabilitating people with disabilities our user base would also expand drastically and we would enjoy more widespread use of our application and could receive grants from investors and possible governmental institutions as well.

Member Responsibilities

As can be seen from the table below, each member was assigned specific tasks based on their area of expertise. The tasks include:

- Research into finding out our targeted user base and the efforts that can be made to effectively engage those users.
- Background research towards existing technology and its problems. This was the first core problem we had to solve in order to figure out which specific use case we had to target and improve.
- Researching alternative multimodal approaches to solve the problem that was being created by the existing technology and whether they were feasible enough to be implemented.
- Selecting a multimodal approach and figuring out what features are implementable. To figure out what features we were going to implement we considered the user base that we were targeting.
- Working on designing the system as follows:
    - Designing the interface for the user to interact with the system based on the principles learned in class
    - Implement modern technologies such as Computer Vision and Speech Recognition to successfully prepare a working model of the system
    - Prepare an application with all the functionalities implemented in it

Members	**Responsibilities**	**Contribution**
Rohit SHYLA KUMAR	Research, programming of computer vision module of the platform and input configuration	100%
Rishabh TANDON	User community research & analysis, possible future extensions of the current design, detailed justification of the design based on the theories, principles, and guidelines, overall report structuring and editing	100%
Sameer KABANI	Research on existing technology and its problems, user interface design and analyzing alternative modal interfaces	100%

Prashant SRIDHAR	Final design research and selection, python programming for the platform and debugging.	100%
Archit CHOUDHURY	Designing of the selection process, UI/UX design, programming of voice recognition module and multithreading.	100%