VOICEHUD

A voice command HUD interface for safer driving

(and less clutter in real estate restricted interfaces)

A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) (Software)

School of Electrical Engineering and Computer Science
University of Newcastle, Australia

Word Count: approx 12000
Date Submitted:

Name: Julius Myszkowski
Email: c3155112@uon.edu.au
Signed: _____

Supervisors: Dr. Shamus Smith
Signed: _____

Dr. Mira Park
Signed: _____

ABSTRACT

Many preventable car accidents are caused by drivers changing their field of view away from the road onto other interfaces and devices. Research shows that current methods of control and locations of these interfaces result in driver distraction. VoiceHud (VH) addresses these problems by providing a minimally distractive voice controlled heads up display (HUD) GUI. User testing was conducted on a prototype using driving simulations, and the results showed that users thought this interface could prevent accidents. VH's can also be an effective interface in situations where voice is a natural mode of communication, such as in collaborative environments, interfacing with a virtual human, or in situations where interface real estate is limited. To demonstrate this, the VoiceHud concept was integrated into the final year group project and user tested with the results confirming the benefits in these situations.

1 INTRODUCTION

1.1 Background and Motivation

Modern devices, control units and traditional dashboard instruments are in locations that distract from essential regular focal areas of safe responsive driving. Since the features they provide are beginning to be replicated into HUD's, these areas are serving as unnecessary distractions. Whenever the driver temporarily moves his/her viewpoint downwards off of the road onto these instruments, GPS, mobile phone or central display unit, the risk of driving unresponsiveness and accident increases.

The road ahead is the driver's immediate focal point and gets the most focus. The driver also scans traffic up ahead, as well as behind and to the side lanes using the rear and side mirrors. Other objects such as street signs on the side of the road, pedestrians near or on the road, and to a lesser extent divergent oncoming traffic on the other side of the road are in constant peripheral vision scan.[1] The behavior of glancing down to the dashboard to get operational feedback (such as speed, fuel remaining and car temperature) was traditionally necessary for driving safely within speed limits and other conditions. Long deviations away from these key focus areas (see appendix 9.1) has been found to contribute to driver distraction.

The introduction of media players, GPS devices, mobile phone devices and central display units has consequently led to more opportunity for driver distraction. Since there is now an increase in trend for both original car manufacturers and after-market products offering HUDs (Heads Up Displays), much of the functionality of these distractions can be incorporated into the HUD to provide the driver with a non-distractive interface. The HUD also allows for augmented information not previously offered such as Advanced Driver Assist.

However, incorporating all of this information into the view of the driver in a HUD interface, along with the problem of multitasking context switch, may also create a risk of distraction. Research is still ongoing in this area for car HUDs, but historically there were cognitive overload problems with plane pilot HUDs.[2][3] Whilst the majority of research suggests that HUDs help with the distraction problem, too much information on the screen can also lead to accidents.[4] Methods of user interaction used in HUDs are likely integral to preserving the benefits of less distraction and more driver responsiveness. Some systems are using touch based menus and some are using air gestures, both contradicting the age old rule that facilitates responsiveness of keeping both hands on the wheel.

VoicehHud (VH) was therefore motivated by the primary goals of keeping the driver's eyes only on the essential focus areas (with the HUD aiming to achieve this), as well as enabling the driver to keep two hands constantly on the wheel (the voice controller aiming to achieve this).

1.2 Project Aims

The primary aim of the project was to develop a system that integrates voice commands with a dynamic HUD display GUI that can be used for multiple implementations and domains and when implemented for the driving domain: (a) keeps the driver's eyes on the road ahead by minimising redundant interfaces and the associated context switch and (b) keeps both the driver's hands on the steering wheel by limiting the user interfaces to a voice controller (and optional steering wheel button controller).

The secondary aim was to Integrate some component or concept of the system into the final year project (FYP) group project. The third aim was to conduct user acceptance testing (UAT) on both implementations (a) the driving simulation and (b) the group nursing project.

The overarching aim of the project was to engineer and document the system using good practices, including designing, methodology, implementing, testing and project management.

1.3 Deliverables

The deliverables of the VoiceHud project included an in-depth literature review on driver distraction, car voice command interfaces, car HUD interfaces and driving simulation; some basic driving task analysis; an implementation of the VoiceHud system including integration in a driving simulation; an implementation and integration of the VoiceHud system into the group project; documentation of the system and method (this document and a user guide); and a conduct and evaluation of user acceptance testing.

2 LITERATURE REVIEW

2.1 Driver distraction

2.1.1 Definition and Summary

It's generally accepted amongst the literature that the two main behaviours that cause road accidents are (a) the driver being distracted from the road and surroundings and (b) the driver taking their hands off the wheel and losing control. This section describes driver distraction. Driver distraction (or inattention) has been defined as "The diversion of attention away from activities critical for safe driving toward a competing activity, which may result in insufficient or no attention to activities critical for safe driving" [5] [6]. Another distinction is that of diverting attention from the driving task to a secondary task which includes more common distractions such as winding down the window or reacting to an unexpected horn. Any new information presented to the driver involves recognition and interpretation and thus has the opportunity to distract from the driving-task.[7] The literature identifies four forms of distraction - visual, auditory, biomechanical and cognitive. A major focus of visual distraction literature and of highest concern to this project is that of glancing off of the road to address some other task or look at some other interface. Here some also make a distinction between driving related inattention (glances onto operational dashboard instruments and mirrors) and non-driving related inattention (glancing onto phones, radio etc) [8]. Whilst reliable measures of visual demand (for example tracking eye movement) have been developed into clear guidelines for reducing visual distractions in vehicles (NHTSA 2012)[8] and are easily measurable with eye movement tracking [9], the difficulties of predicting and measuring cognitive attention has made reliable measures harder to develop and standardise.
It has also been found that repeated quick back and forth glances onto some other interface is more detrimental than a single long glance of the same period [10]. Research on cognitive distraction suggests that even if a driver's eyes remain on the forward roadway, his or her ability to detect and respond to targets within the visual field may be impaired if cognitive (thought) focus is not also on the forward roadway [11] [12]. This phenomenon is more broadly known as inattentional blindness [13]. Conversation with passengers is the most frequent secondary task followed by eating, smoking, manipulating controls, reaching inside the vehicle, and cell phone use. A significant proportion of the existing literature is devoted to assessing the impact of cell phone use on driving performance and safety. Although cell phone use represents a relatively small part of the overall distraction problem, use among drivers is steadily growing with approximately 10% of drivers using some type of cell phone at any point in time. Although not representative of the U.S. experience, the available evidence suggests that cell phone use increases drivers' crash risk by a factor of 4. [81]

2.1.2 Dashboard Instruments and Controls

Dashboard instruments are traditionally situated behind the driving wheel for quick glancing of operational instruments (including fuel gauge, odometer, speedometer, car temperature). These give important information needed for safe driving. However because of their location, they still contribute to glancing off of the road and down onto the instruments and thus contribute to visual and cognitive distractions as the driver looks and interprets the information. This is also the case for adjusting controls like temperature and fan controls, which are to various extents necessary tasks whilst driving, but even when designed with the intent of ease of use, still distract away from the main driving-task.

Fig 1.1 Dashboard instruments behind steering wheel.[16]
Fig.1.2 Modern cars use digital representation.[17]
Fig.1.3 Radio [18]

2.1.3 Music, Music Players and Digital Clocks

Both background music and music player control units are another source of driver distraction [14] [15]. Music players often result in the driver taking eyes off the road and a hand off the wheel. Often in the same area, digital clocks also result in glances off the road.

2.1.4 GPS Devices

Jensen et al, 2010 [19] studied how different output configurations (audio, visual and audio-visual) of a GPS system affect driving behaviour and performance. They conducted field experiments in real traffic and found that GPS devices not only cause a substantial amount of eye glances, but also led to a decrease in driving performance - measured by longitudinal control error (speeding violations), lateral control error (lane excursions), and eye glance behaviour. Adding audio output decreased the number of eye glances, but saw no significant effects on driving performance. Although the audio configuration implied much fewer eye glances and improved driving performance, several participants expressed preference for the audio/visual output.

Many other studies have looked at on-road GPS and map interfaces and some guidelines have been developed for designing them to be safe (see Appendix 7.4) [20] [21] [22].

2.1.5 Mobile Phones

The research into mobile phones show that they are a major driving distraction.
Strayer et al 2003 [23] found that cell phone use impairs driver stopping ability in simulated driving likely because of impaired object recognition memory when focus is distracted onto conversation. Strayer and Drews 2007 [24] examined the effects of hands-free cell-phone conversations on simulated driving and found that even when participants looked directly at objects in the driving environment, they were less likely to create a durable memory of those objects if they were conversing on a cell phone. This was evident for objects of both high and low relevance, suggesting that very little semantic analysis of the objects occurs outside the restricted focus of attention. They also found that in-vehicle conversations don't interfere with driving as much as cell-phone conversations do, because drivers are better able to delay difficult driving tasks in in-vehicle conversations than with cell-phone conversations.

Cuřín et al 2011 [25] compared various forms of text input including a non-visual version as well as cell phone texting and GPS entry as a baseline. They found that the eyes-free version keeps the distraction level and other versions caused more distraction than the eyes-free version and were comparable to the GPS entry task. The cell phone texting task was the most distracting one.

2.1.6 HDDs

Some modern cars are using central display units to replace traditional controllers for music player, temperature control, gps etc. These are also termed in the literature as Heads Down Displays (HDDs) and a number of studies have found that HUDs result in less visual distraction off of the road than HDDs. Liu [26] compared HUDs and HDDs under both low and high driving load conditions, and found that drivers reacted faster to speed limit sign changes than when paying attention to the road. Jæger et al [27] also found that a touch based HDD was less distractive than tactile or swipe gesture based HDD. Keifler [28] compared visual sampling behaviour and speed control performance during daytime driving with both HUD and HDD (see sections on HUDs) speedometers and found that HUDs were less distractive.

2.2 In-car Voice command systems

Research into in-car voice command systems is still only about a decade old and not considered complete, but studies showing both positive and negative outcomes have been conducted. A few studies caution that auditory-vocal tasks may lead to unexpected task demands [29] [30] [31]. Vollrath 2008 [32] compared speech vs manual operation and concluded voice operation keeps eyes on the road and that the main requirement for design of in-vehicle information systems (IVIS) is sole voice operation for both safety and acceptance. Pengjun 2008 [33] found intuitive voice based systems led to less driving errors. Strayer et al 2013, 2014 [34] studies' suggest that cognitive demands associated with speech-to-text system interactions may be higher than other common auditory-vocal tasks such as talking on a cell phone or listening to the radio. Cooper, Ingebretson & Straya (2014) [35] then measured cognitive demands in six different voice controlled systems by periodically instructing participants to perform interactions such as; dial a 10 digit number, call a contact, change the radio station, or play a CD - all using "hands-free" voice systems. The voice systems were activated by touching a button on the steering wheel. Mental workload was derived by measuring reaction time, subjective assessments, and heart rate and they found the following results. Firstly, the most critical element of mental workload appeared to be the duration of the interaction, of which the primary contributing factors were the number of steps required to complete the task as well as the number of comprehension errors that arose during the interaction. Secondly, common voice tasks are generally more demanding than natural conversations, listening to the radio, or listening to a book on tape. Thirdly, if designed well, these same basic commands can be completed with little error in very few steps, leading to little additional cognitive demand. Fourthly, speech production whilst driving is significantly more demanding than speech comprehension. They also found that system errors increase cognitive workload. Finally, they found that less steps to perform tasks suggested less demanding workload

2.3 HUDs (Heads Up Displays)

Whilst traditionally, dashboard instruments, music players, GPS, HDDs and mobile phones were necessary to serve different goals (see section 2.1.3-2.1.6), this information is now being replicated into HUDs. The idea for HUDs originated from the military and aviation industries where they were used for delivering key information into the pilot's view without lag (but not without problems [36]). These days they are being used more commonly in passenger planes and military fighter jets. The first commercial car HUD was used in 1988 but the idea has seen an increase in research and products due to an increased focus on driver safety as well as increase in technologies (see appendix

2.3.1 Benefit

A number of studies have shown benefits of HUDs in presenting operational information over HDDs. Sojourmer et al 1990 [37] found that subjects operating a driving simulator with a HUD speedometer reacted significantly quicker to salient cues in the driving scene versus a typical dashboard-mounted speedometer. Burnett 2003 [38] found that participants made fewer navigational errors (wrong turns) when an arrow graphic and contextual information about the surrounding roads was presented on a HUD versus on a HDD. Charissis et al 2008 [39] used a simulator to test an HUD-based collision warning system in low-visibility conditions (typical of heavy rain or fog) and found it dramatically reduced the number of collisions and improved subjects' maintenance of following distance, when compared to a traditional HDD. Liu and Wen 2004 [40] compared HUD and HDD in commercial trucks and found that HUD was superior for conveying emergency-related information (e.g., pedestrian warnings and engine temperature status). Weinberg et al 2011 [41] tested street name finding using HDD, HUD or audible only displays and found the auditory display had the least impact on driving performance and mental load, but at the expense of completion efficiency. The HUD variant had a low impact on mental load and scored highest in user satisfaction (see figures 2.1).

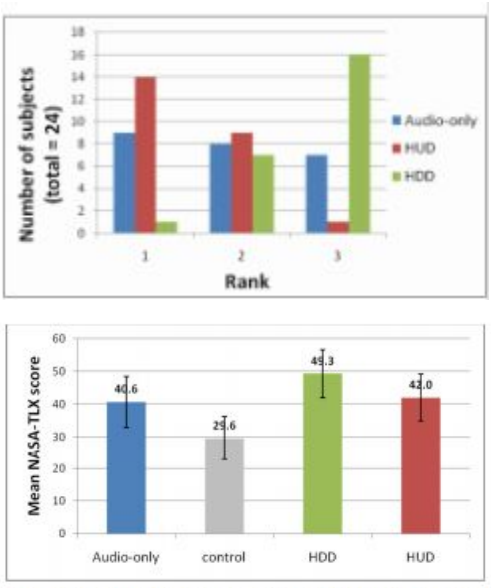


Fig. 2.1 (top) User interaction preference in Weinberg 2011 (above) Mean workload for these interactions.

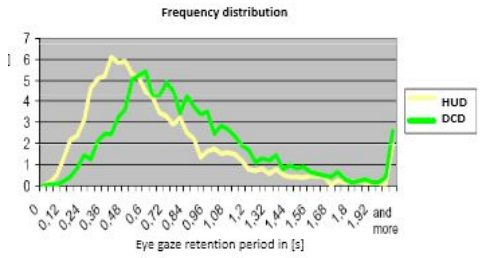


Fig.2.2 Bengler found HUD had less gaze retention.

Bengler [42] 2007 compared HUDs with HDDs and found 85% of users accepted and desired

the HUD while driving. This result was independent of age, system experience, and domain-specific knowledge (see figure 2.2).

2.3.2 Design Considerations

2.3.2.1 Ufov

Miura (1990) [43] defined the Useful field of view (UFOV) as the visual area over which information can be extracted at a brief glance without eye or head movements. UFOV has been found to decrease with age, most likely due to decreases in visual processing speed, reduced attentional resources, and less ability to ignore distracting information. UFOV performance is correlated with risk of an accident. UFOV can be tested using computer simulations reliably and performance can be improved with computer based training [44].

2.3.2.2 Guidelines

Fumero 2004 [45] suggested the following guidelines for IVIS (In-vehicle information systems):

1. information should be presented auditorily to the drivers.
2. information should not be presented with only visual displays.
3. if a visual display is used it must be in a multi-modal display where audio is used as well.
4. auditory displays can use either normal or fast speech rates to present verbal information.
5. Multi-modal displays should use a normal speech rate to present verbal information.
6. systems where tasks can be performed with minimal hierarchical steps.
   Others have provided more guidelines (see Appendix 7.4)[46][47].

2.3.2.3 Text, Font and Location

Watanabe et al. 1999 [48] studied the effect of HUD warning location on driver responses and found that drivers did not like locations that were farther than 5 degrees horizontally from the center and preferred locations that did not overlap with locations where important events occurred (e.g., in the exact center).Tsimhoni et al. 2001 [49] studied the best position for presenting short text messages on a full-windshield HUD, evaluated in terms of driving performance and workload. They confirmed that message locations within 5° of a straight-ahead gaze yielded the best performance and were preferred by subjects. These conclusions were based on very short messages only. This was also the case for simple navigation instructions.

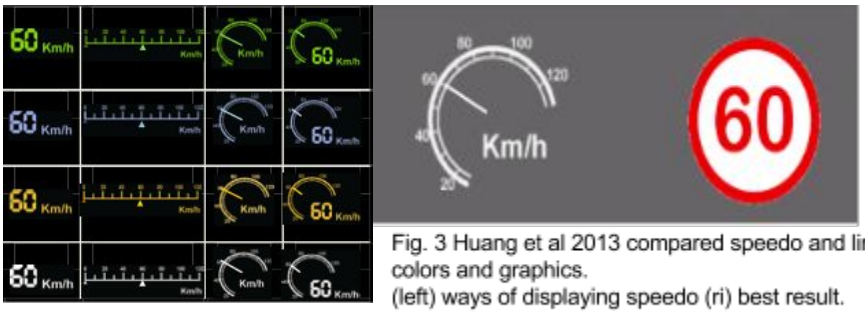


Fig. 3 Huang et al 2013 compared speedo and limit colors and graphics.
(left) ways of displaying speedo (ri) best result.

2.3.2.3 Color, Icons and Graphics

Conflicting colors could prevent drivers from clearly seeing information and result in an accident instead. Huang et al 2013 [50] studied effects of HUD interface design and found the best average stability was achieved with a low transparency, outlined speed limit, white & round speedometer. They also suggest use of only static and not dynamic images.

2.3.2.4 Timing

Timing of information is important to consider in order to not overload or confuse the workload of the driver. Campbell 1998 [51] gives specific event timing information. Most of the guidelines suggest that information should not distract the driver for more than 2 seconds.

2.3.3 Advanced Driver Assistance

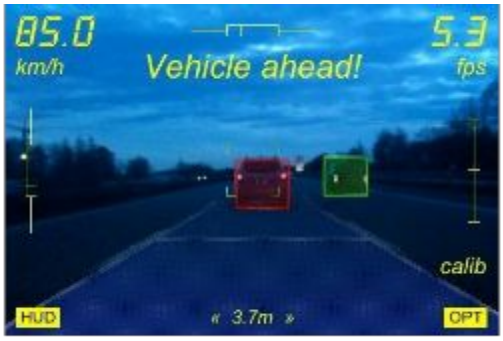


Fig. 4 Example of object detection/NVES in HUD [70]

2.3.3.1 Object Detection and Night Vision

Keifer and Raymond 1998 [52] found that HUD based pedestrian detection was beneficial in preventing accidents for older drivers. Driving at night time is more difficult than driving in the day due to lower lighting visibility.Rumar 2002 [53] looked at NVES and found most of the research is in favor of it.

2.3.4 Steering Wheel Touch Controllers

Steering wheel located touch or tactile controllers can be used as both redundant and alternative controllers to voice controlling of HUDs. This is a better method than air gestures since it keeps both hands on the wheel. Murer 2012 [54] used this approach and saw good initial results.

Fig. 5 Steering wheel thumb controller Murer 2012

2.4 Driving Simulation

Simulation, controlled and real-world tests are all useful for user testing tasks performed on any car based interface. Whilst simulation and controlled driving can facilitate greater experimental control, the latter better simulates real driving conditions and situations, this is even more true of real-world testing. However controlled and real-world driving tests have both practical and ethical restrictions. Bach et al 2008 [55] found both simulation and controlled driving tests lead to similar results, however, they yield more frequent and longer eye glances compared to simulated driving and driving errors were more common in simulated driving.

2.4.1 Measuring Visual Load and Distraction

Visual distraction and demand can be measured by eye tracking using infrared cameras. Whilst traditionally such a setup was expensive, cheaper options are now available including virtual reality devices such as the Oculus Rift [56].

2.4.2 Measuring Cognitive Load and Distraction

The Lane Change Test (LCT) is an easy-to-implement, low-cost methodology for the evaluation of the distraction associated with performing in-vehicle tasks whilst driving [57][58][59]. The test subject drives on a three lane road, where signs continuously indicate various lane changes to be carried out as quickly as possible. The test measures how well the driver follows a specified ideal path. To measure and compare the level of distraction caused by various secondary tasks, each task is carried out while driving. Typically, increased distraction results in a corresponding decrease in driving performance. This test was developed by vehicle manufacturers and is currently advocated as an international standard (ISO/DIS 26022) [60] to record standardised distraction effects. The lane change test has been used to assess distraction tests of visual-manual and speech-based operation of navigation system interfaces and to show the effectiveness of HUDs at response preparation [61][62].

2.4.3 Simulators

SImulating driving allows for road driving software systems to be tested in a controlled environment with no risk of accident and harm to the users. Simulators at a basic level can be mocked, or can be realistic and aim to replicate the real driving experience. One such realistic, open source computer based driving simulator is OpenDS (Open Driving Simulator)[63][64].

3 DESIGN AND METHODOLOGY

3.1 Design

3.1.1 System requirements

The system was designed to meet the following requirements.

3.1.1.1 Functional Requirements

F1. The system must provide voice command recognition.
F2. Voice commands must be recognisable in real time.
F3. The system must provide a HUD interface.
F4. The system must include text to speech capability for speech responses to the user.
F5. The system must route voice commands to specific views of the HUD interface.
F6. The system must be provide functionality of key tasks provided by other interfaces.
F7. The system must include a logger for auditing voice commands and other data.

3.1.1.2 Non functional Requirements

NF1. The HUD interface must only show a maximum of 3 view elements at once, so as not to incur too much cognitive load on the user.
NF2. The HUD interface must not be too distractive from the driving task.
NF3. The HUD interface must not be too cluttered.
NF4. Textual information on the HUD should be limited.
NF5. Where possible information should be delivered to the user as speech.
NF6. Where speech is not intuitive, the HUD should utilise visual information before textual information where this is intuitive.
NF7. The system should be simple and easy to use so the user would use it over other interfaces.
NF8. The voice command shouldn't chain commands deeper than 3 levels so as to prevent cognitive demand.

3.1.2 VoiceHud Use case Diagram

Figure 6.1 shows the basic uses of a generic VH.
The speaker can either speak a voice command or show or hide the HUD.
An auditor can view logs of commands made and views showed.

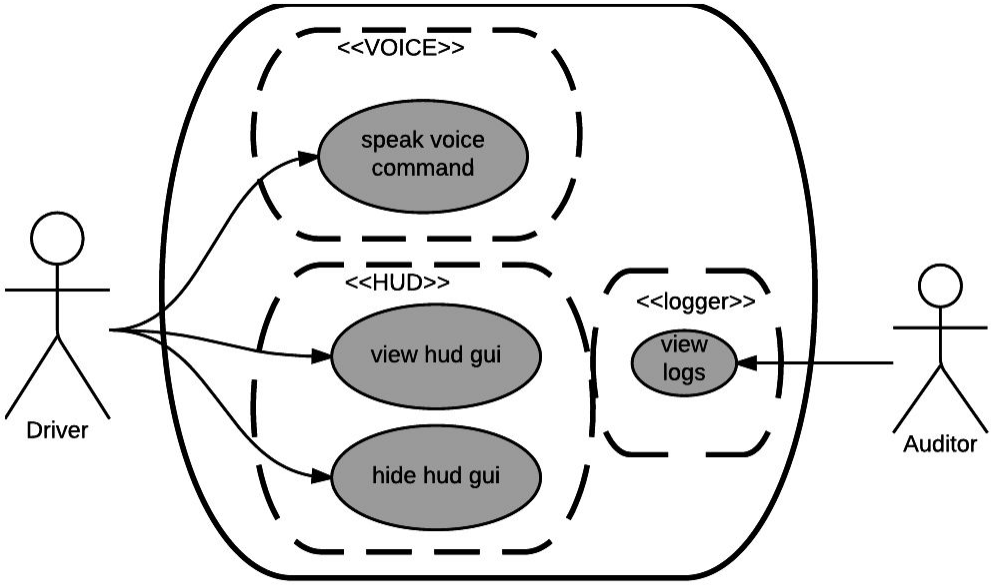


Figure 6. VoiceHud Use Case Diagram

3.1.3 Task Analysis

From the (literature in section 2) the major secondary tasks performed whilst driving are:

conversation with passengers eating smoking reaching inside the vehicle	manipulating controls cell phone use

Scoping out those on the left column that are less relevant to a VoiceHUD interface, the major use case areas to focus on for the VoiceHud driver implementation to minimise distraction from other interfaces become those on the right column.
As detailed in section 2, for critical feedback data such as speed and speed limit, immediate, responsive minimal text in the driver's useful field of view is less distracting than a voice readout (that takes longer for cognitive processing), therefore viewing the current speed and speed limit become critical use cases. On the other hand, for non critical tasks, to achieve less distraction, cognitive demand and clutter, a voice readback of the data combined with simple uncluttered GUI elements is optimal. Thus, reading out a text message is prioritised over displaying a whole message. Elements that require a simple measure are used with voice readout. Using a simple Hierarchical Task Analysis (HTA) tree, the use cases for the VH driver system are identified along with the critical (red/orange), noncritical (green) and out-scoped (gray) tasks.

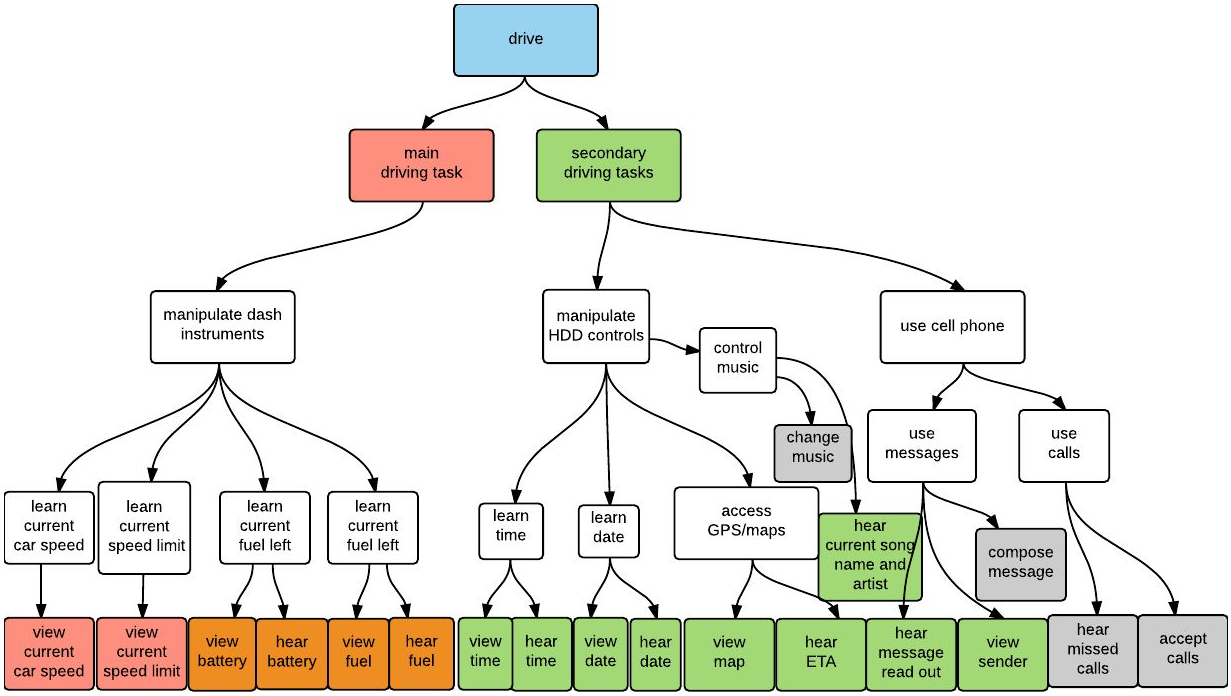


Figure 7. HTA of Driving Tasks (red/orange critical tasks), green (secondary), gray (outscoped)

3.1.4 VoiceHud (Driver) Use case Diagram

The main use cases identified in the HTA become the use cases of the VH driver system. These are generally described by the user either commanding the system to access various information or responses or the user viewing or hearing these responses.

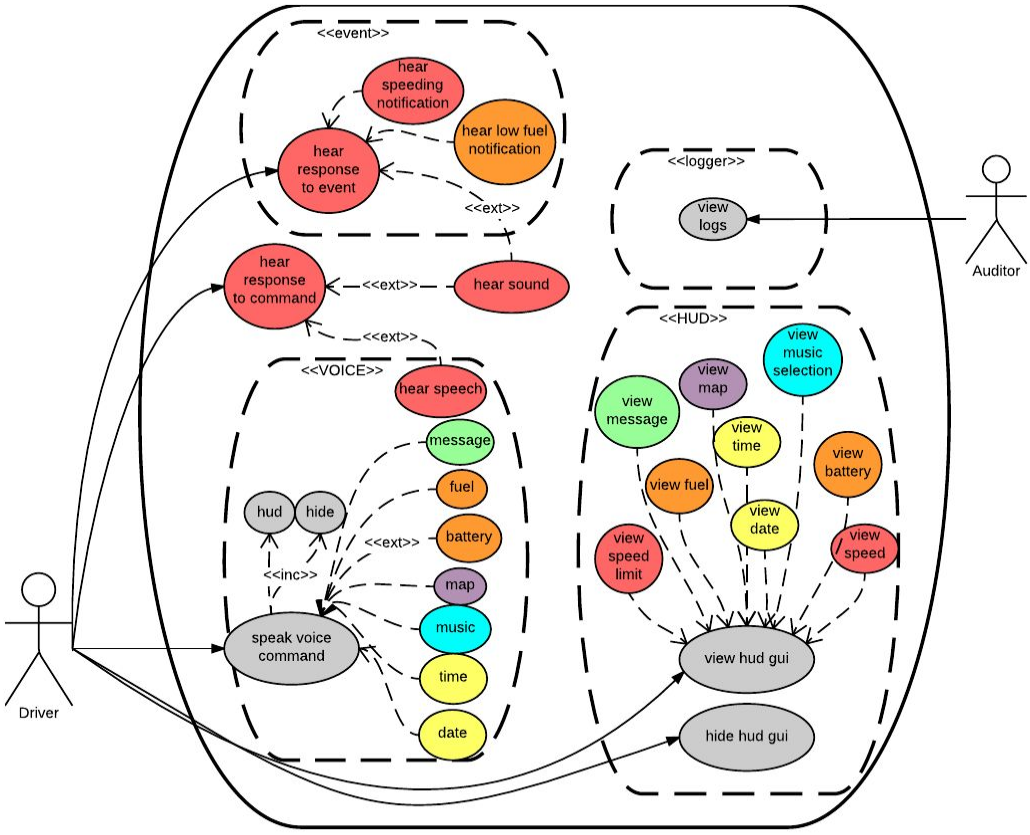


Figure 8. VoiceHud (Driver) Use Case Diagram

3.1.5 Use Case Descriptions

	Use Case	User	Description
U1	voice command “hud”	driver	command shows the main HUD view
U2	voice command “fuel”	driver	command enables user to learn fuel left
U3	voice command “battery”	driver	command enables user to learn battery left
U4	voice command “messages”	driver	command enables user to access recent text messages
U5	voice command “map”	driver	command enables user to see map
U6	voice command “music”	driver	command enables user to learn current song name and artist playing
U7	voice command “time”	driver	command enables user to learn current time
U8	voice command “date”	driver	command enables user to learn current date
U9	voice command “hide”	driver	command hides the main HUD view

Table 1. VoiceHud (Driver) Use Case Descriptions

3.1.8 HUD View Controllers

3.1.8.1 Critical HUD View Controllers

From the use cases of section 3.1.7 the view controllers needed of the system were designed.

	View	Description of controller function	Use Case
V1	HUD with current speed and speed limit	The main HUD controller. For the driving simulation this shows the main hud view speed and current speed limit, since these things are essential to the driving task.	U1, U9
V2	Fuel	Displays the car's remaining fuel level (%).	U2
V3	Battery	Displays the car's remaining battery level (%).	U3

Table 2.1 VoiceHud (Driver) Critical HUD View Controllers

3.1.8.2 Other HUD GUI View Controllers

	View	Description of controller function	Use Case
V4	messages	message view displays senders photo and reads message	U4
V5	map	The main HUD view. For the driving simulation this shows the speed and current speed limit, since these things are essential to the driving task.	U5
V6	music	Reads out current song and artist playing.	U6
V7	clock / time	Displays current local time.	U7
V8	date	Displays current local date.	U8

Table 2.2 VoiceHud (Driver) Non-Critical HUD View Controllers

3.1.10 System Overview

The voicehud system is designed to be an extensible and reusable system that routes voice commands mappings to gui views. Any domain specific information can be used by either integrating the VH core components C1-C4 (as well as C5-C6 if needed) into an existing system or by implementing the simulation component C7. The focus of this submission is for a driver simulation but other systems such as for the nursing or medical domain could use it too.
The system is also designed to have decoupled components and to be stateless, but a layer of state persistence could always be added should the implementation need this.

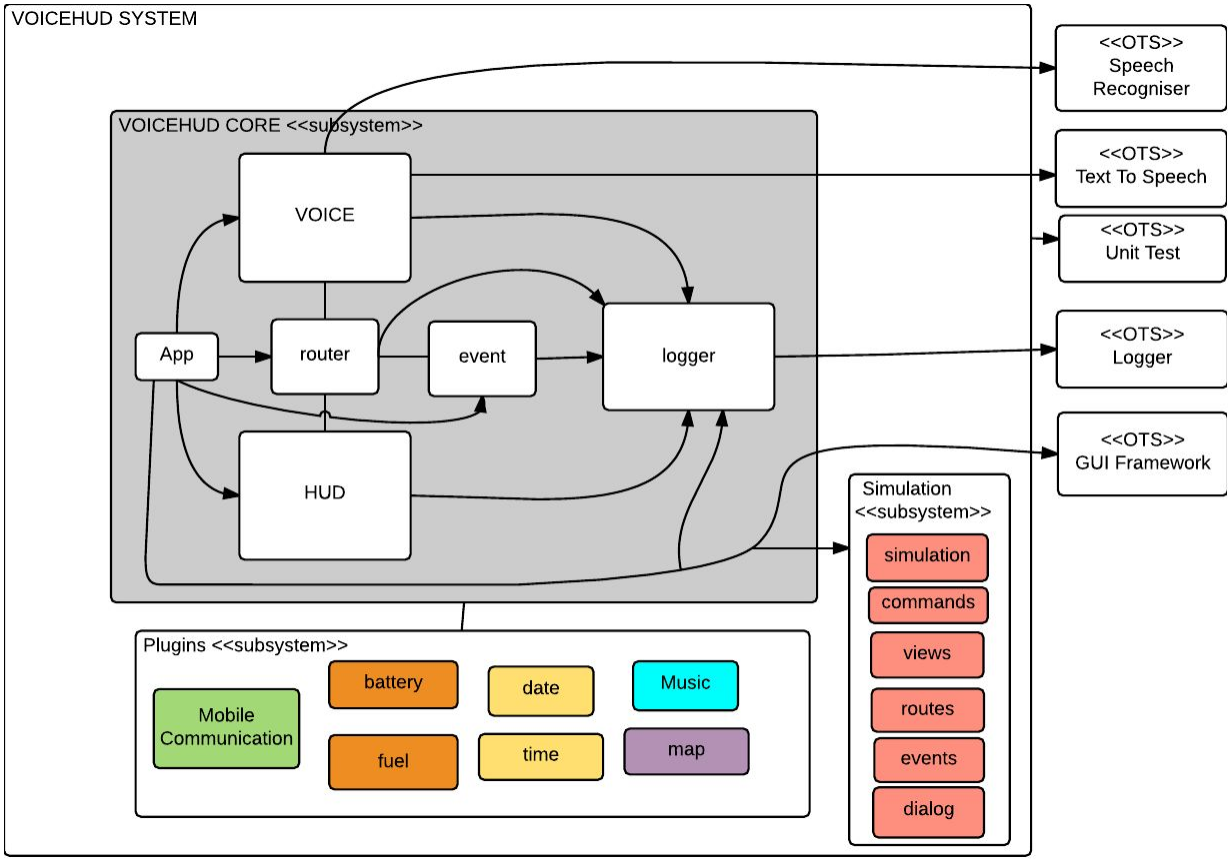


Figure 9. VoiceHud System Overview Diagram

3.1.9 Voicehud Core Subsystem

The core subsystem of VH consists of four main components, a *voice* component, a *hud* component, a *router* component and a *logger* component. There is an additional three components for implementing a voicehud for a specific domain and simulation.

3.1.9.1 App Component (C1)

The app component is the component responsible for setting up and managing all the other components.

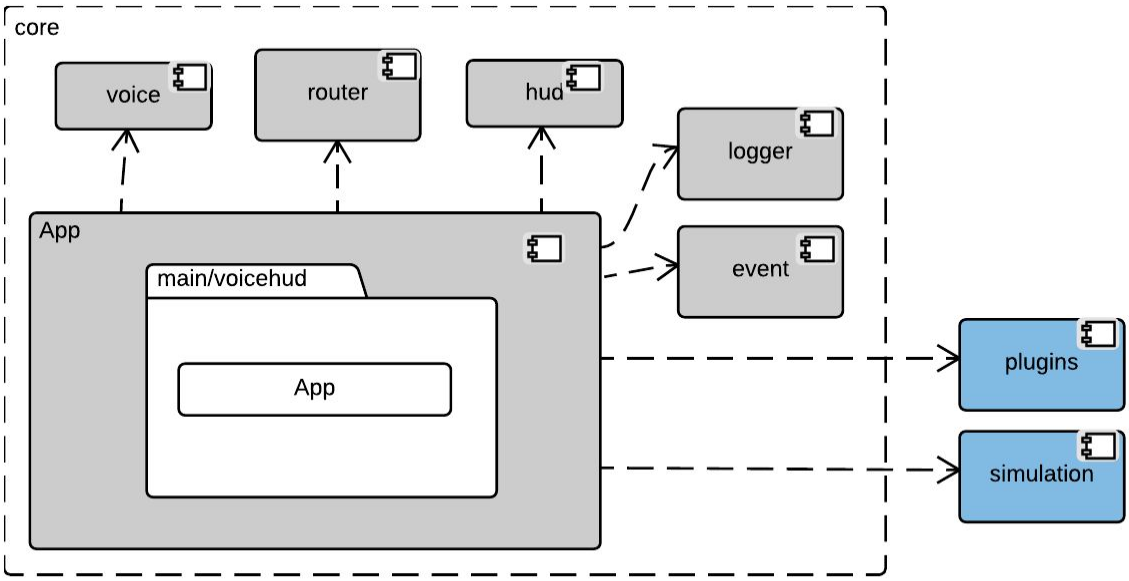


Figure 10. App Component Module Diagram

3.1.9.2 Voice Component (C2)

The voice component is responsible for recognising spoken words and sentences, primarily commands, but is extendable to recognising running speech for dictation for example. This component could also be extended (with a voice authentication library) to recognise user voice signatures but that is out of the scope of this submission.

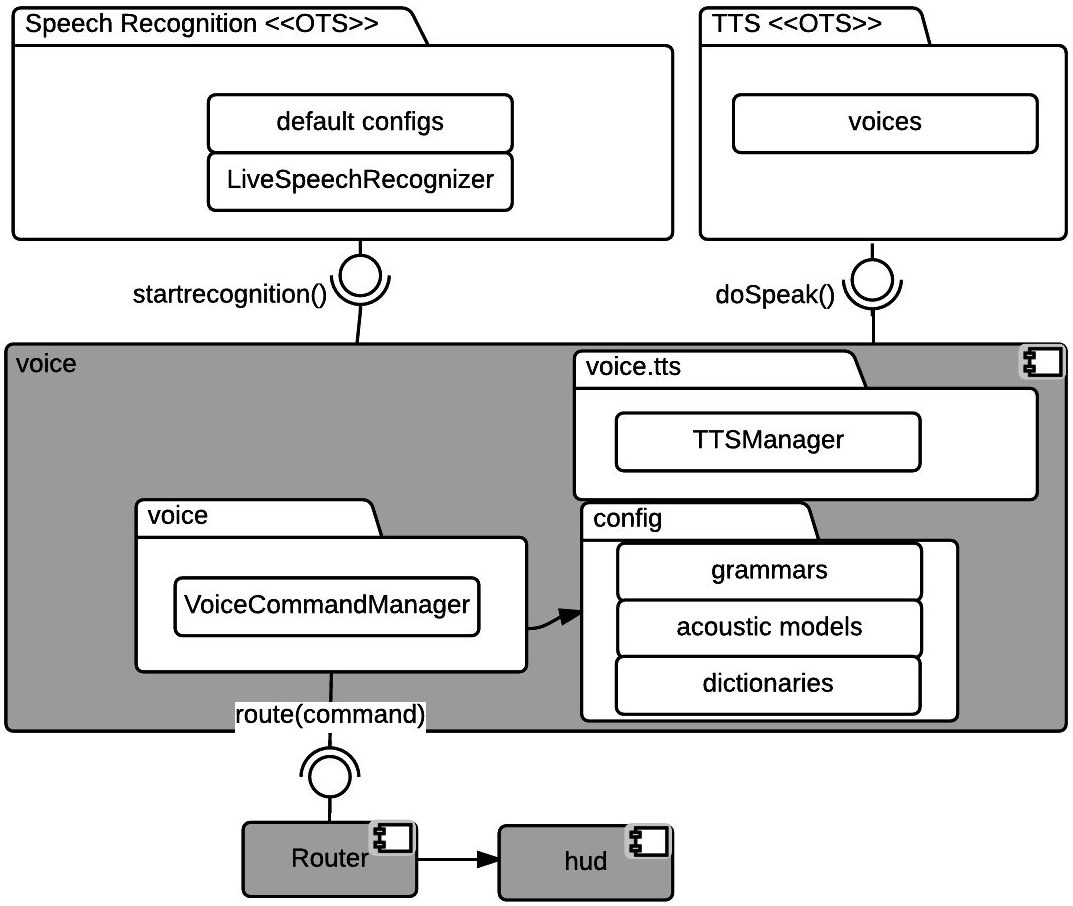


Figure 11. Voice Component Module Diagram

3.1.9.3 HUD Component (C3)

The HUD component is responsible for displaying the HUD GUI to the user. This includes graphics and textual elements. These elements are grouped together as views. Views can include multiple layers.

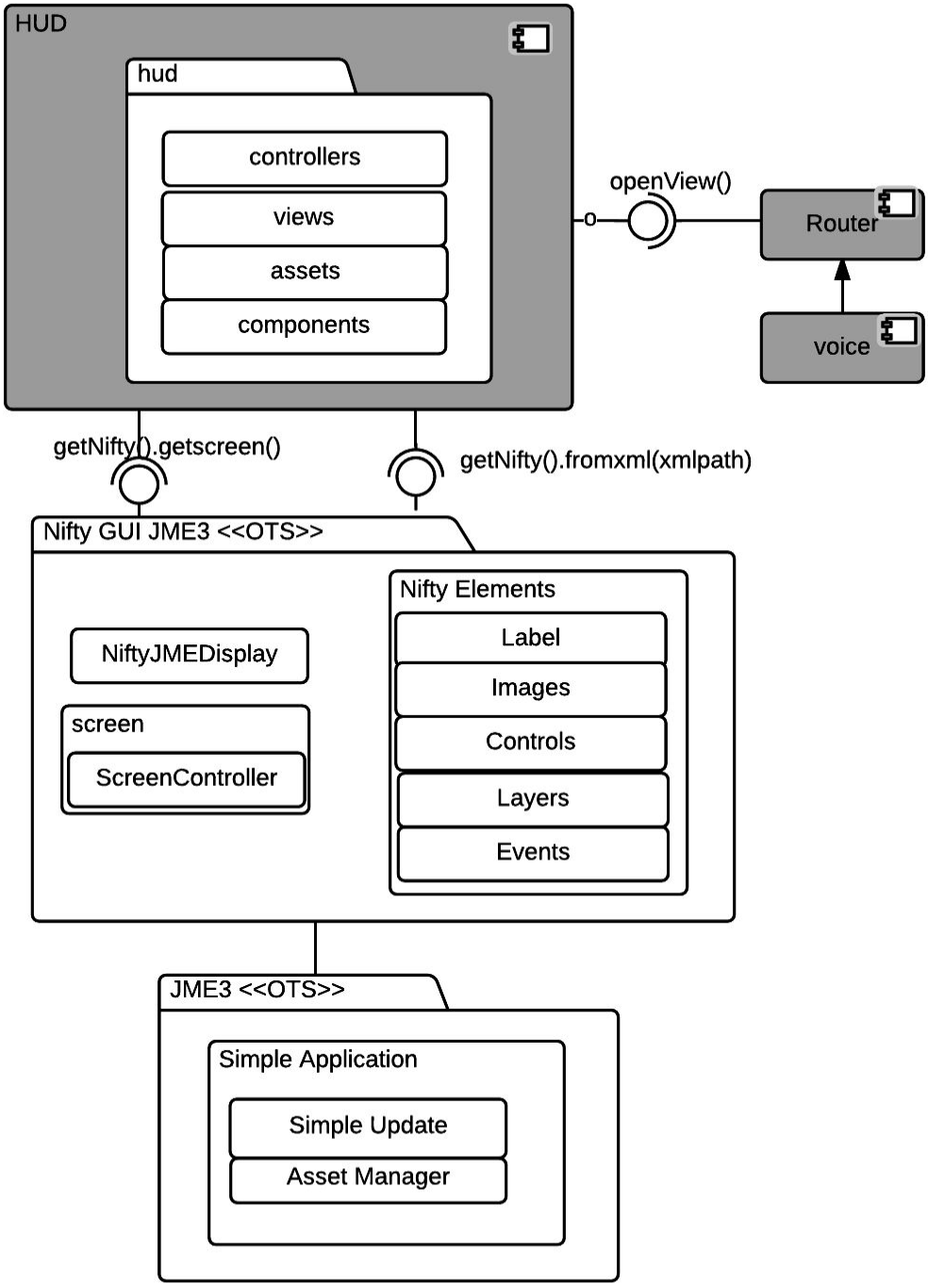


Figure 12. HUD Component Module Diagram

3.1.9.4 Router Component (C4)

The router component is responsible for routing voice commands to HUD GUI views. There also might be situations where events route specific views, especially in other implementations.

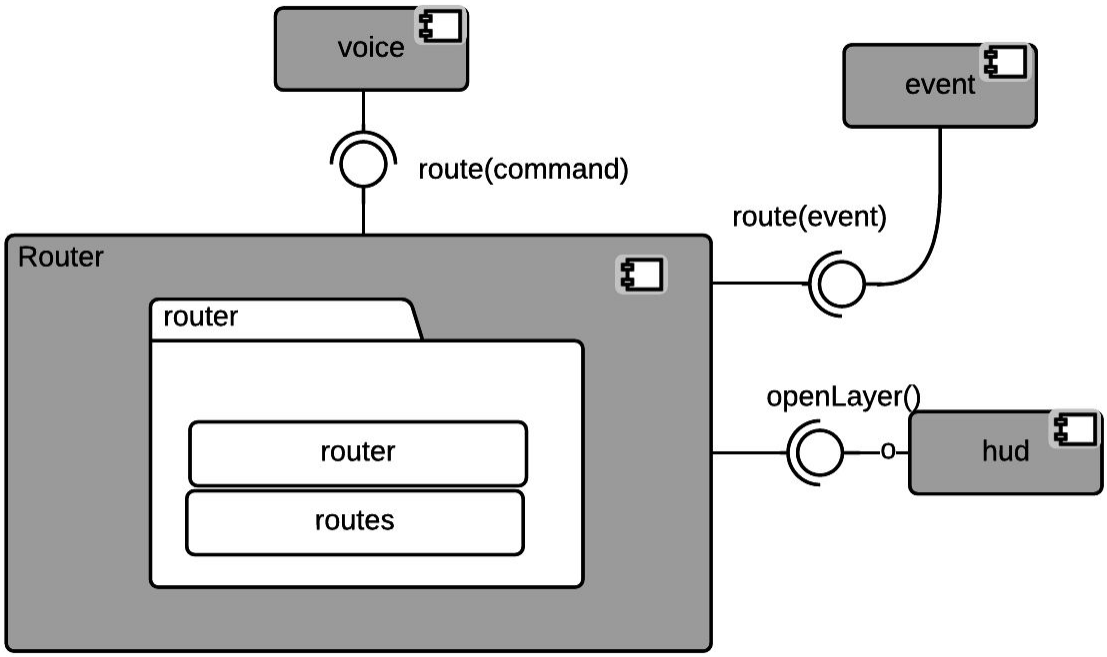


Figure 13. Router Component Module Diagram

3.1.9.5 Event Component (C5)

The previous three components map the user's voice commands to different views using a router. We can have the system to read text for example when a gui view loads. But the system may also need to show content to the user at certain times or in certain situations not based on GUI view loads. For example, when the driver is speeding, the GUI should be alerted. Thus, the event component is responsible for acting on certain triggers. Events could include audible and visual alerts to the user. In the context of the group project, an example of an event could be when the user accomplishes a milestone and some feedback is provided using the HUD. Of course this is extra functionality to the base VH requirements, but is essential in the driving implementation for real use, as well as it would be for many other real applications and so is included in the core subsystem for extensibility and decoupling from the HUD's view controllers. This component would deal with certain events but not define them.

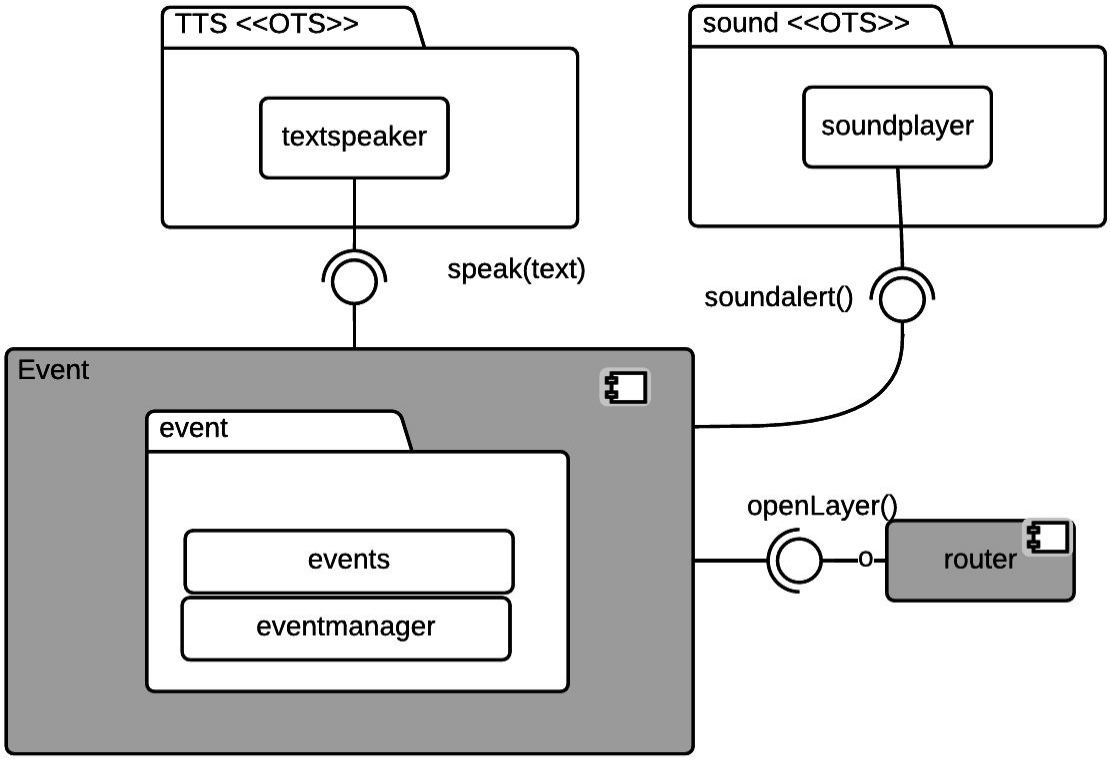


Figure 14. Event Component Module Diagram

3.1.9.6 Logger Component (C6)

The logger component is responsible for logging data between components such that an auditor could inspect them at a later date.

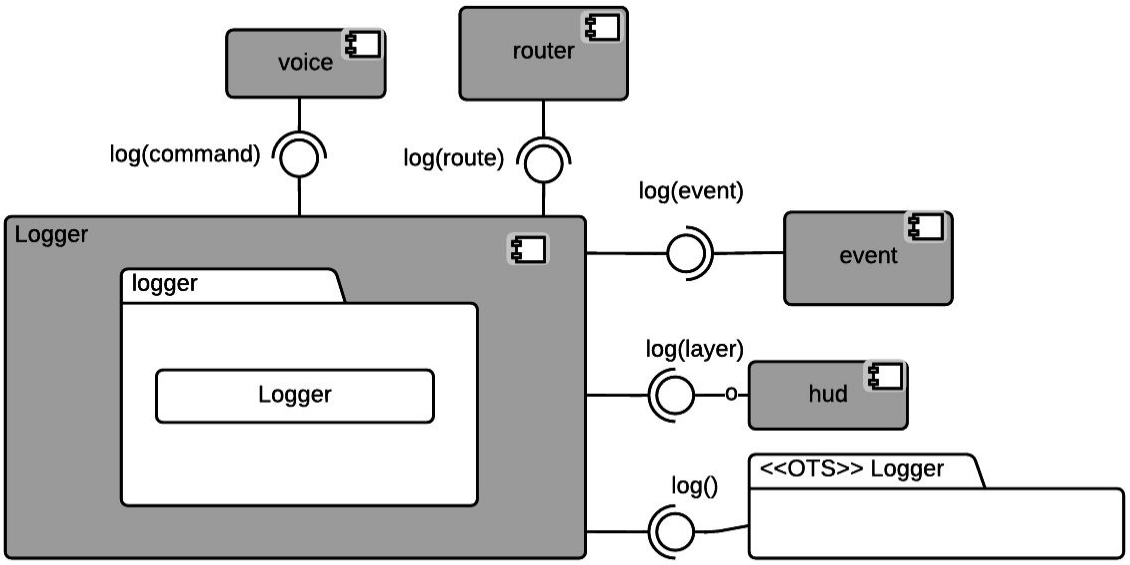


Figure 15. Logger Component Module Diagram

3.1.10 Plugins Subsystem

The plugins subsystem contains plugins that are not core to the voicehud system but add additional feature functionality. These plugins aim to be modular and potentially reusable across simulations and implementations. For example, the music component could be used by different implementations of VH.

	Plugin	Description	View
P1	Mobile Communication (messages, calls)	Manages phone calls and text messages. Can be extended to utilise existing OS APIs (Android, IPhone) or a custom or mocked API. In another domain, the message system could be useful to access messages from other users.	V4
P2	Maps	Manages maps and GPS. Can be extended to utilise existing GPS APIs.	V5
P3	Music	Manages music player. Can be extended to utilise existing music APIs (Spotify, Apple Music, Google Play etc) but this submission uses a mock API.	V6
P4	Battery	Displays and reads out battery remaining.	
P5	Fuel	Displays and reads out fuel remaining.	V2
P6	Date	Displays and reads out current date.	V3
P7	Time	Displays and reads out current time.	V8
P8	OBD (outscoped)	Manages information gathered from the OBD2 input of the car. Under certain circumstances will trigger events to be managed by the event component.	NA
P9	Object Detection (outscoped)	Detects objects using input from cameras, including forward, side and back facing. Has different detectors for other cars, humans and other objects. Under certain circumstances will trigger events to be managed by the event component.	NA

Table 3. Mapping plugins to view controllers

3.1.11 Simulation Subsystem

3.1.11.1 Simulation Component (C7)

The simulation subsystem abstracts the implementation of the routes, logic, commands and views such that any simulation in any domain can implement a VH simulation. This includes a simple simulation implemented with a movie player. This simple simulation was used by the basic VoiceHud driver implementation.

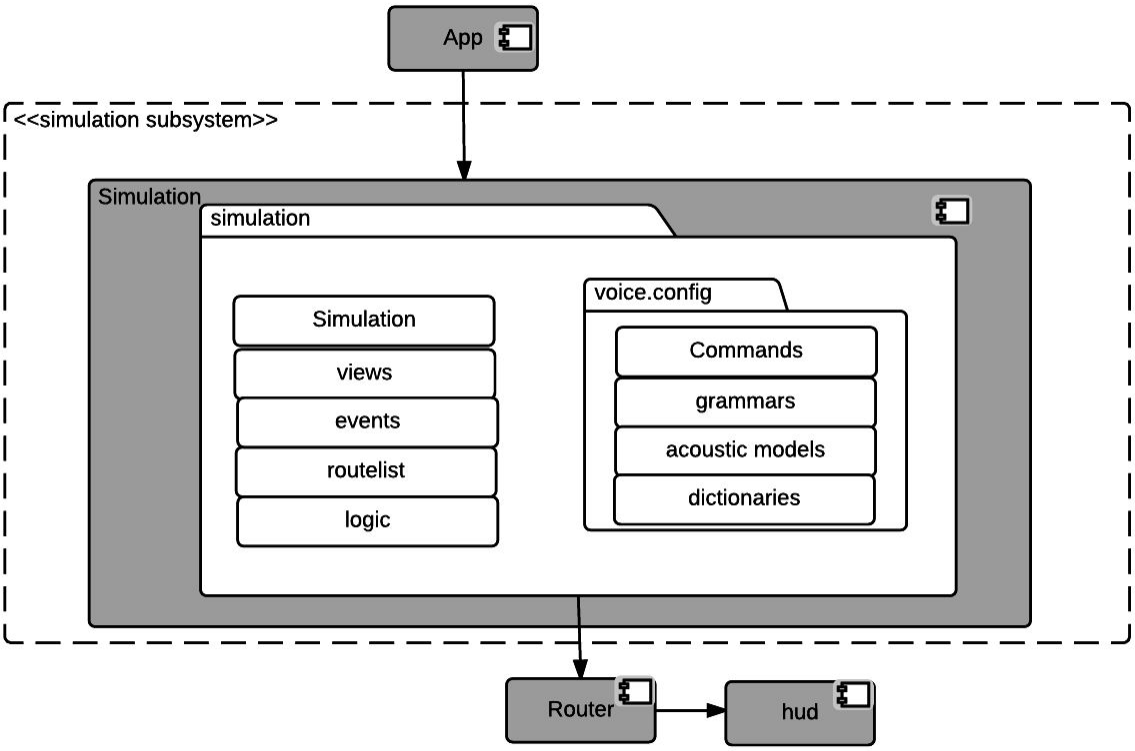


Figure 16. SImulation Component Module Diagram

3.1.12 User Interface Design

A number of views were mocked up using Photoshop in order to experiment with what the GUI would look like.

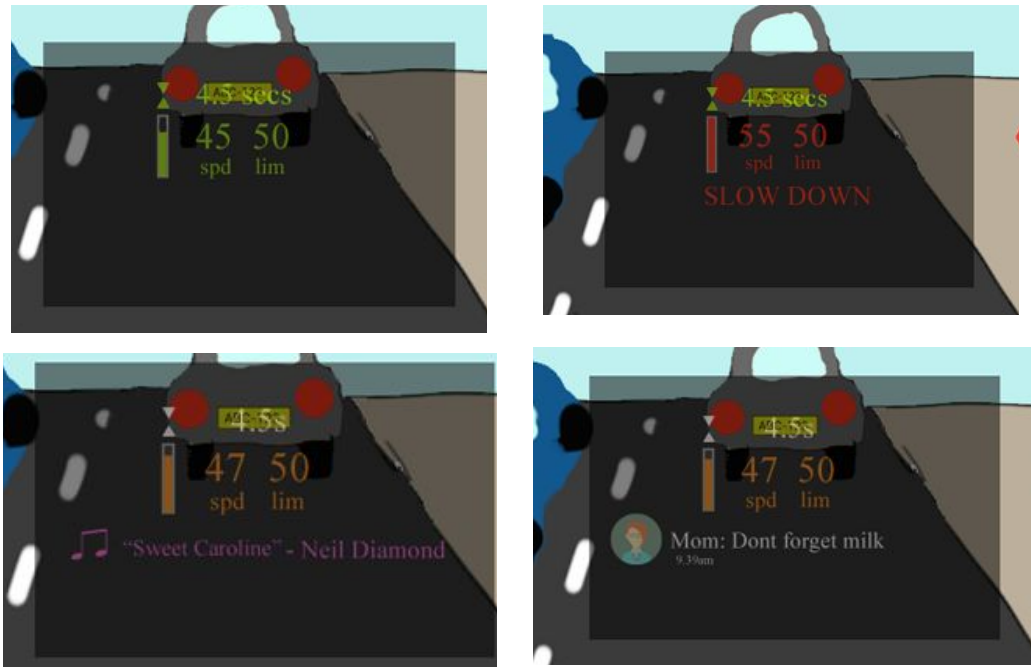


Figure 17. A number of mockups of HUD GUI views produced using Photoshop.

3.2 Methodology

3.2.1 Development Method and Tools

Development was conducted in an iterative agile manner since the requirements evolved quite rapidly. Three iterations of the project were planned as deliverables, a basic, intermediate and advanced version.

Eclipse IDE was used as many dependencies (eg. OpenDS) were well documented for Eclipse setup.

Github was used for source repository with Git version control because it is free and easy to use. The git repository was located at (www.github.com/jskye/voicehud). Different branches were used for each the basic to advanced deliverables above.

Trello was used to keep track of task management and github issues were used to keep track of issues because these tools were light weight and thus easy for one developer to manage.

Maven was used for build control and dependency management allowing easy integration of dependencies and specification of builds including test builds and the ability to manage different deployments in the future (out of scope of this submission).

A windows installed machine was used for testing setup of the openDS integrated voicehud. See appendix 9.8 for the driving simulation setup.

The overarching design pattern used in development was of view controllers. The views and controllers are decoupled such that a view can easily be designed in xml and the controller can then handle update logic.The xml interfaces utilize the element tree hierarchical structure of nifty and jme. A model layer could easily be added where implementations require it but it's not included in the system by default since VH is fundamentally a controller (voice) interface (hud).

3.2.2 Testing

3.2.2.1 User Testing

The VoiceHud system was tested using firstly a basic video playback simulation and then secondly a simulated Lane Change Test in the openDS driving simulator. Users were then asked to fill out a customised System Usability Survey (see Appendix 9.7-9.9) on their user experience.
Four systems were planned to be tested using the openDS driving:

1. The VH system.
2. The VH system without voice control and with steering wheel buttons toggling views.
3. The voice controller by itself, with sound feedback but no HUD GUI.
4. A baseline without the VoiceHUD system.

However, due to time constraints, only one basic test (1) with the user utilising the VH whilst performing a lane change was conducted. The plan was also to measure the distraction from the driving reports that the openDS provides. But due to configuration error these were deemed unreliable and the SUS only used.

3.2.2.2 System and Unit Testing

Unit tests were conducted on the core VoiceHud components to ensure a working core.
Since the other systems were loosely coupled from the core, so long as the core passed testing, then the system was somewhat trusted, and other subsystems would be isolated when the need arose. Whilst TDD was not used for this project, a discipline of unit testing after major refactors and component additions was adopted. JUnit was used for unit testing java classes. Each component was set to have its own test suite, such that if any specific test in that suite fails, then the component's suite fails.

```
// A test method to test the 12 hr time method.
// twelve hour time is default since 00:12 read as number is confusing.
      @Test
      public void isTwelveHourTime(){
             // mock the time method but in 24 hr time.
             Calendar cal = Calendar.getInstance();
             SimpleDateFormat sdf24 = new SimpleDateFormat("hh:mm");
             String militaryTime = sdf24.format(cal.getTime());
             Assert.assertNotEquals(dummyClockLabel, militaryTime);
      }
```

```
// The HUDTests suite runs tests on many controllers.
     @RunWith(Suite.class)
     @Suite.SuiteClasses({
              TestClockController.class,
              TestDateController.class,
              TestMessagesController.class,
              TestHUDGUIController.class
              })

// The RunTests class runs a suite of suite runners.
@RunWith(Suite.class)
@Suite.SuiteClasses({
        TestApp.class,
        HUDTests.class,
        RouterTests.class,
        VoiceTests.class
        })
```

Whilst the latest version of JUnit supports concurrent testing of threads, this was largely left beyond the scope of development but that version of JUnit is used as the dependency to support such testing.

Whilst much of the HUD GUI was defined in xml, given that Eclipse flags xml if it doesn't validate this was considered sufficient, especially whilst conforming to the nifty XSD. An XMLUnit library was considered but deemed unnecessary.

3.2.2.3 Performance Testing

The system was evaluated prior to user testing for performance measures such as responsiveness, reliability and efficiency so the results would be valid. Only a stable version of the system was used for user testing, where all unit tests passed, despite it being less advanced feature wise.

4 PROJECT MANAGEMENT

4.1 Scheduling & Gantt Chart

Due to the iterative nature of the project and synchronisation to the uni semester a Gantt chart was drawn up in a waterfall like fashion with iterations targeted by sprints to achieve the version milestones with a flexible week should the schedule get behind. These key task deliverables were then added to a Trello Kanban board with appropriate deadlines.

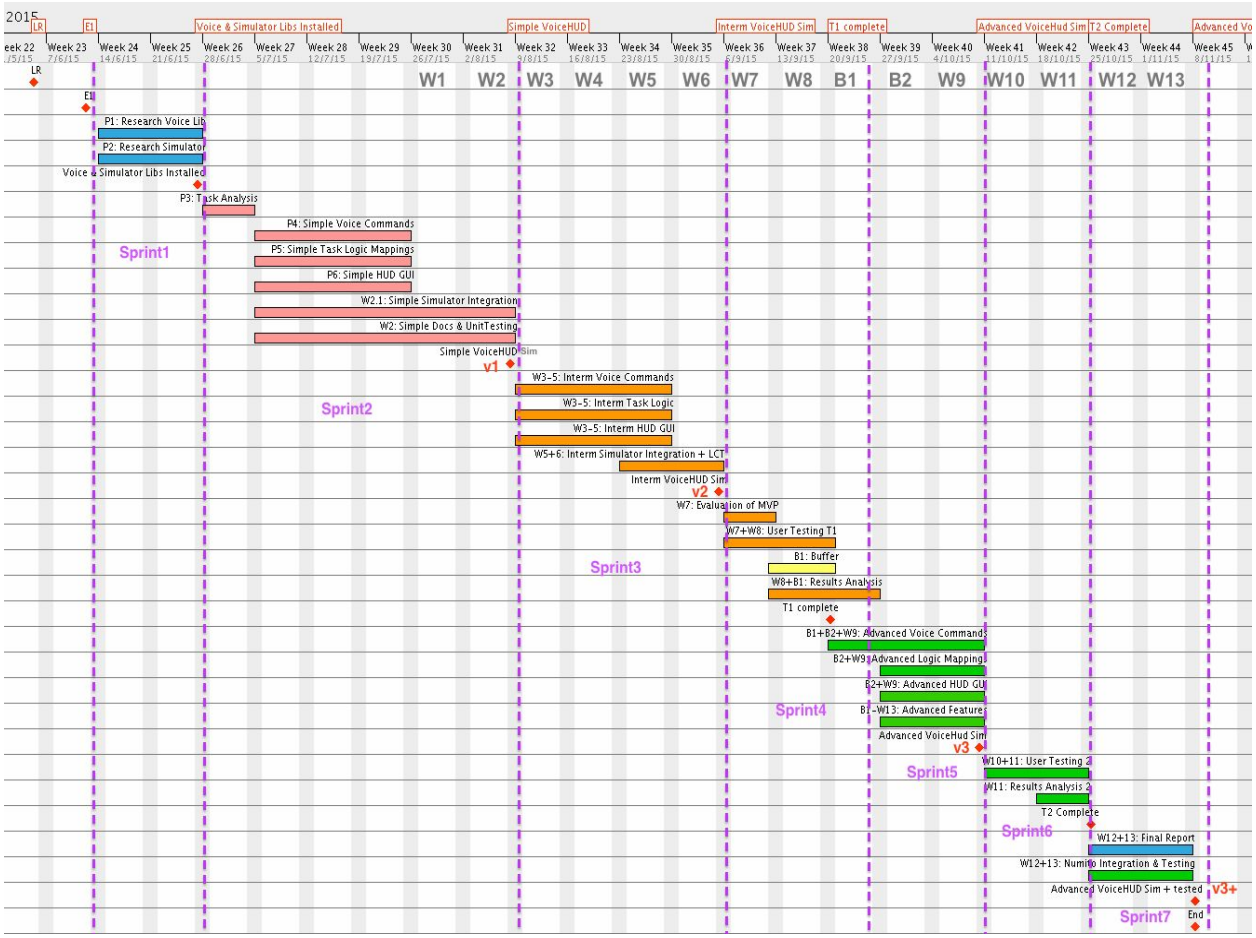


Fig. 18.0 VoiceHUD Project Gantt Chart

4.2 Project Resources

Resource	Cost / Availability
Voice Recogniser - Sphinx4 (and annyang.js)	Free OS OTS
Driving Simulator - OpenDS	Free OS OTS
User Testing Equipment - Windows machine - driving gaming controllers - gaming microphone	 Free (university access) Obtained for free from a friend (RRP ~ AU$300) $60
User Testing Participants	University students
	Total: $60 ($360)

Table 4. Resources, Cost, Availability

4.3 Ethical Considerations

The main ethical concerns of this project were with user testing. University and participant approval was required to get users to test the system. Test date was sanitised for sensitivity.

There is also the concern of quality of the system should it be used upon completion of this project. The system should be accompanied with appropriate licensing, usage warnings, disclaimers and instructions.

To avoid these risks, the primary user testers were other software engineering students from our cohort.

Due to the VoiceHud-OpenDS-plugin integration involving simulated driving, with the potential for simulated crashes, the users were warned of this prior to user testing.

4.4 Risk Management

A risk management plan was drafted at the outset of project development. The main risk identified was of the project not completing fully or in time. This was mitigated by scheduling and versioning as described above. There was also the risk the software won't fix the problem, this is mitigated by research and user testing. Also users may not accept the software - this is mitigated by multiple phases of user testing. There are other user testing risks since real life actions are being simulated, these can be mitigated by preparing the participants before hand for possible outcomes. There is also software risk relying on dependencies like testing env's. This is mitigated by having backup options. In an individual project, there is risk of project stall if developer is incapacitated or resources are unavailable. PM mitigates these.

5 IMPLEMENTATION AND EVALUATION

5.1 Implementation

5.1.1 Summary

Whilst VH is designed to be simulation extensible, the implementation for this project focused on the prototyping for use in driving cars. Whilst, the initial prototype is a simple JME3 application, targeting it to an Android application is very easy to do with the JME framework, which makes it more easy to deploy to an android OS with access to other apps and API's. Whilst the application can be targeted to other domains, they may need additional integration or development of features beyond which VH provides, such as the addition of click and touch events for the multi-touch learning interface, which would be a matter of implementing the relevant click event listeners. Alternatively, VH could be integrated into existing frameworks.

The implementation of VH involved implementing and integrating two core components, one which was an implementation of a JME3 application and Nifty GUI (described in sections 5.1.4-5.1.5) and the other an implementation of Sphinx4 voice recognition (described in section 5.1.6). A router routes the voice commands to relevant GUI views (described in section 5.1.7). A logging component allows the other components to log data.

The class diagram in Figure 13.1 shows that VH is split into core and non core packages. These packages map to the components in section 3.1.11.

From a high level look at the classes, the app class instantiates and initializes the screen states (start screen state and HUD GUI state) as well as all the component managers: VoiceCommandManager, Router, DrivingSimulation and EventManager.

The VoiceCommandManager uses a sphinx voice recogniser to recognise voice commands and sends these commands to the router to route to views. The Router routes the commands to views by looking up its route list and calls the HUDGUIState's open view method which opens the specified view which is implemented in xml. These views bind to controllers that run the logic and update listeners. When certain textual elements load in the views, the TTS speak is invoked to speak out text. The driving simulation implements simulation and holds a simulation route list which implements the router route list.

5.1.2 Class Diagram

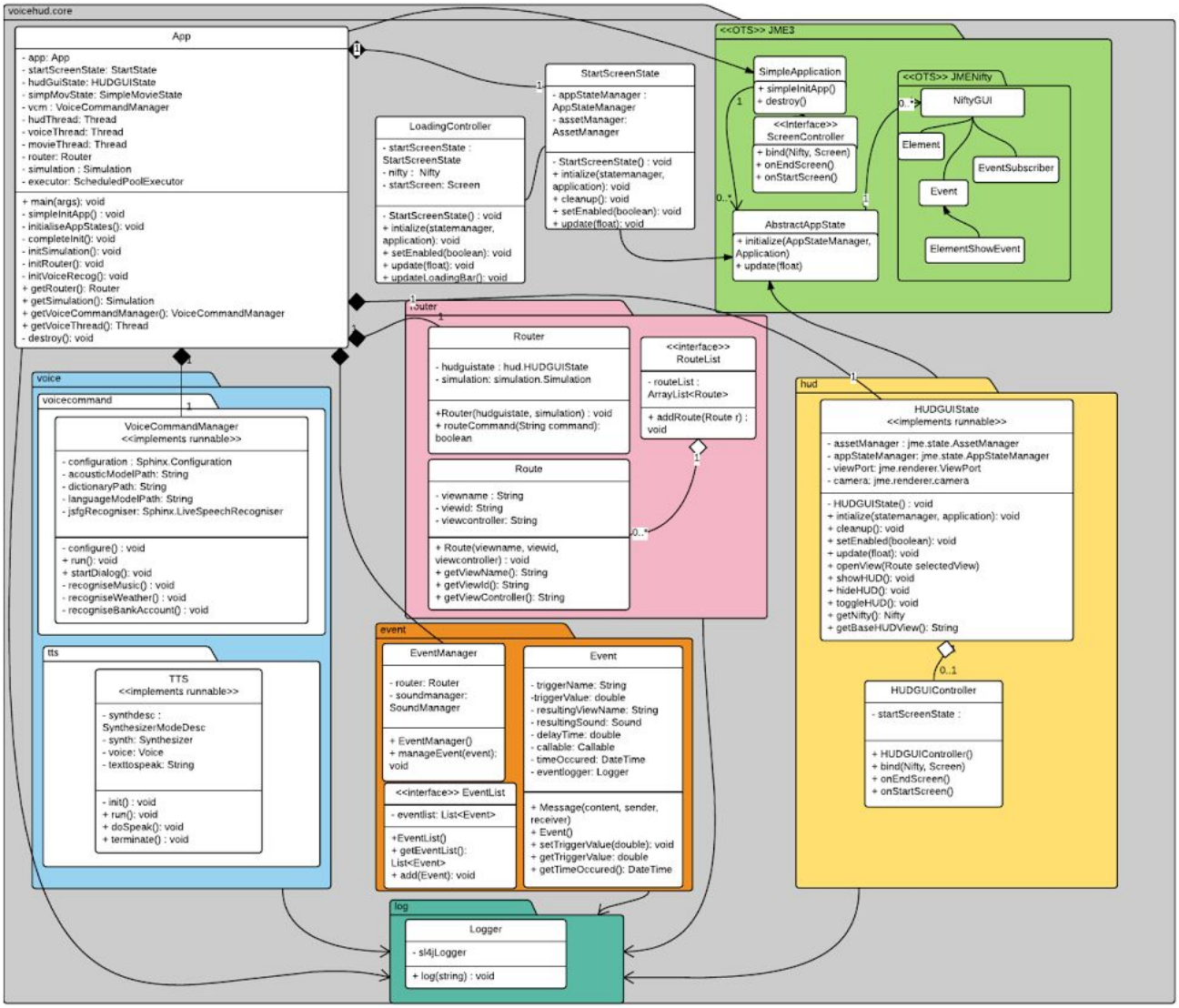


Figure 19.1 VoiceHud Core Class Diagram

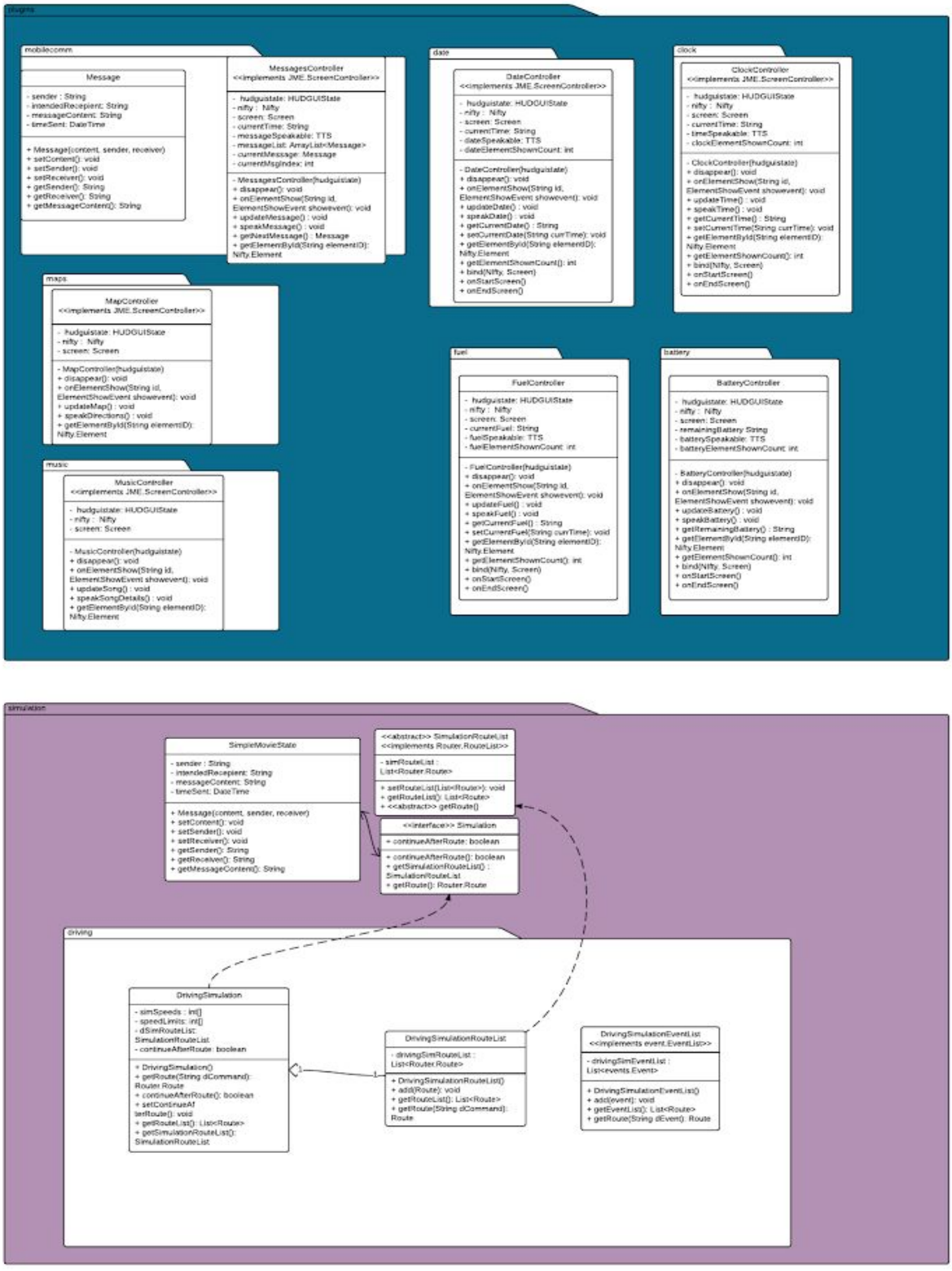


Figure 19.2 VoiceHud Plugins and Simulation Class Diagrams

5.1.3 VoiceHUD Activity diagram

The activity diagram below shows the main flow of states when VoiceHud runs. The program initiates a start screen state that implements a JME AbstractScreenState. This displays a loading screen to the user whilst two other threads do work to initialise. The voice recognition initialises its configuration (and in the driver implementation the simulation's SimpleMovieState initialises its configuration). If a user tries to issue a command before the voice recognition has initialised nothing will happen because the recogniser is not yet listening. The SimpleMovieState also does not begin playing until the recognition is ready. When the recogniser is ready to listen, it calls the movie state to start and then users can make commands whilst the movie plays. VoiceHud at a basic level

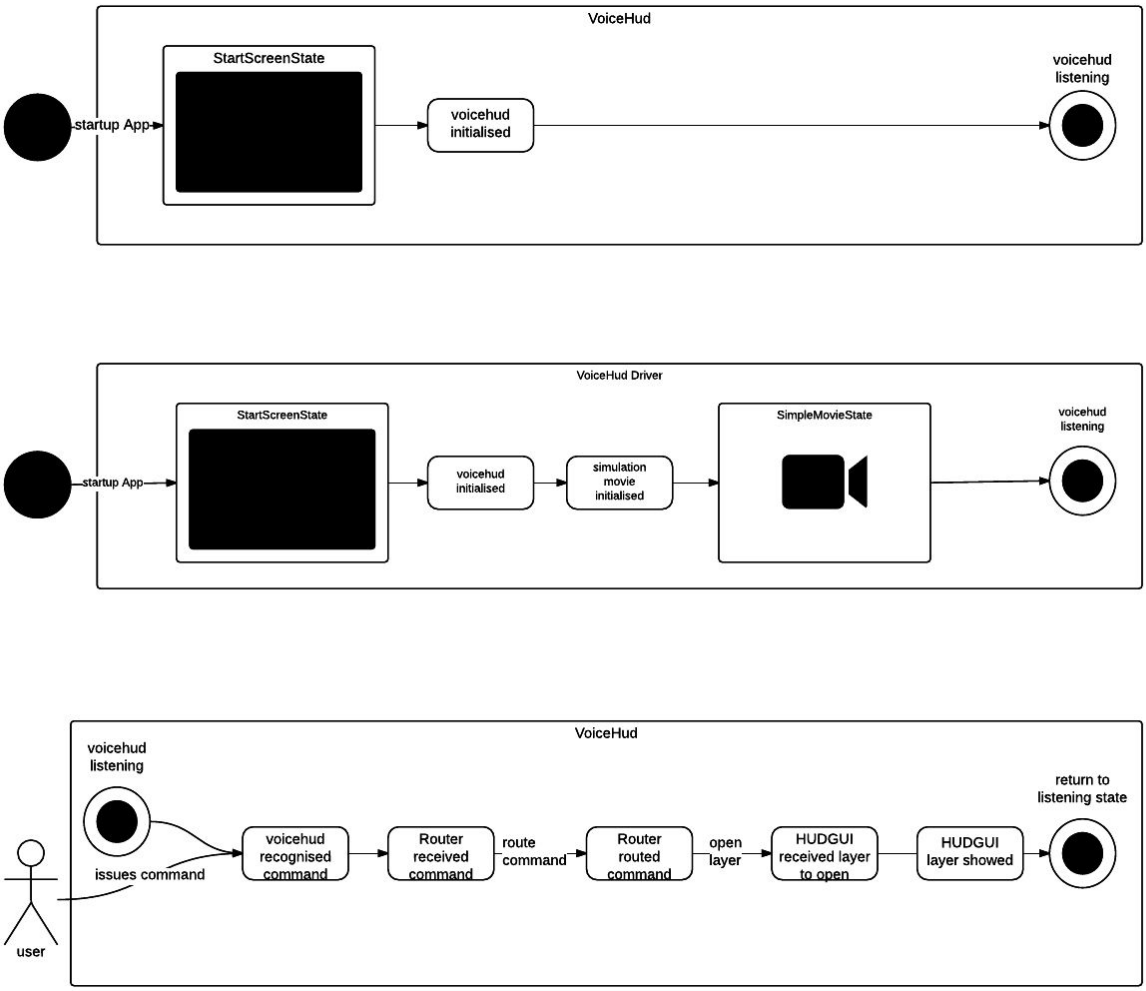


Figure 20. Activity Diagram showing
(top) VoiceHud (general) Setup
(middle) VoiceHud (driver) setup
(bottom) main generic VoiceHud loop

5.1.4 HUD GUI with JME3 & Nifty

Java Monkey Engine 3 (JME3) is a cross platform java game development framework with various features including the ability to target development and deployment to Android OS. For these reasons, as well as the fact that the group project was initially using this framework, JME3 was chosen to form the basis of the HUD GUI. Nifty is a well-supported GUI plugin for JME3 and facilitated easy XML specification of views bound to screen controllers which listen to the main game loop. The Nifty GUI viewport uses screens, layers and panels. The simple VoiceHud project creates and adds views as layers to a default screen controller. Only one screen is necessary here, as layers are added and shown based on the required view. The default layer is a HUD layer. The default HUD layer shows the speed and speed limit at all times such that this information is always available to the driver (see images in section 5.18). Other xml files along with their bound java controllers implement the views designed in section 3.1.8.

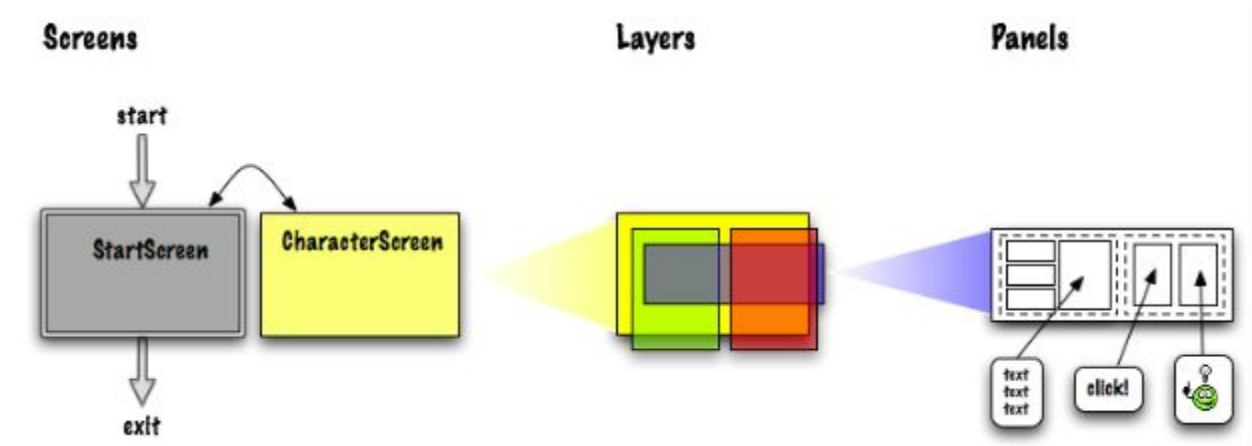


Figure 21. A conceptual view of JME and Nifty
(left) JME screens, (middle) a collection of layers make up views,
(right) panels make up the elements of a layer.
these can be controlled by the bound controller and element updates listened to.

A Nifty GUI Screen forms a View (with layers and panels) in XML:

```
<!-- CLOCK view -->
<nifty ...
<screen id="start" controller="julius.sky.voicehud.core.hud.ClockController" width="60%"
height="60%">
        <layer id="HUD" childLayout="center" backgroundColor="#0000" visible="true"
width="60%" height="60%">
        <panel childLayout="vertical" width="50%" height="100px" backgroundColor="#0f08">
        </panel>
        </layer>
        <layer id="CLOCK" childLayout="center" backgroundColor="#0000" visible="true"
width="60%" height="60%">
                        <control name="label" id = "clockLabel1" text="time" color="#0f88"
align="center" valign="center" height="50px" >     </control>
                </layer>
        </screen>
</nifty>
```

Controls allow for abstracting components, access to a set of default nifty elements and for dynamic manipulation of information via the view controller. The main update loop implements app states that have access to the application's assets and other fields and can control viewports. Nifty is implemented as a GUI viewport. The view's controller class implements a nifty ScreenController which hooks it up to the view.
It also extends AbstractAppState which gives it access to the game application and the main update loop.

```
import de.lessvoid.nifty.screen.ScreenController
public class HUDGUIState extends AbstractAppState ...
public class HUDGUIController implements ScreenController ...
```

In the clock view example above we can easily update the label control (with default "time" text) in the controller logic to show the current time whenever this view is shown.
We can also add various nifty event listeners that subscribe to on an event bus.
For example, this method is notified every time the element with id = "clock", that is, the clock view is loaded.

```
@NiftyEventSubscriber(id="CLOCK")
        public void onElementShow(final String id, ElementShowEvent showevent ) {
                updateTime();
                speakTime();
        }
```

5.1.5 View Controller Mappings

View	View XML Class	Controller Class
V1 HUD	HUD_VIEW	HUDGUIController
V2 fuel	FUEL_VIEW	FuelController
V3 battery	BATTERY_VIEW	BatteryController
V4 messages	MESSAGES_VIEW	MessagesController
V5 map	MAP_VIEW	MapController
V6 music	MUSIC_VIEW	MusicController
V7 clock	CLOCK_VIEW	ClockController
V8 date	DATE_VIEW	DateController

Table 5. Mapping designed views to implemented xml views and their controller classes.

5.1.5 Voice Command with Sphinx4

Sphinx4 is an open source, modular, flexible and easy to set up speech recognition framework developed by Carnegie Mellon University, Mitsubishi, Hewlett Packard and Sun Microsystems. It is written in java and is noted as state of the art as well as better for specific grammar, dictionary and language and acoustic model requirements. CMU Pocketsphinx is a lighter weight and faster implementation. For this project, Sphinx4 was chosen for its ease of setup, java base, and portability, however, due to the real time requirement of the driving deployment, pocketsphinx would be investigated for deployment for the use of real cars (out of the scope of this project). [65, 66]

Sphinx facilitates easy specification of a grammar in Java Speech Grammar Format (JSGF):

```
#JSGF V1.0;

grammar dialog;

<menu_command> = hud  | messages | clock | time | date | hide     | fuel   | map ;

public <command> = <menu_command>;
```

We then use the LiveSpeechRecognizer interface to recognise speech, which gets a hypothesis of what the recogniser think it heard which we can then base our logic on:

```
        Configuration configuration = new Configuration();
        configuration.setAcousticModelPath(ACOUSTIC_MODEL);
        configuration.setDictionaryPath(DICTIONARY);
        configuration.setGrammarPath(GRAMMAR);

        LiveSpeechRecognizer jsgfRecognizer =
            new LiveSpeechRecognizer(configuration);

        jsgfRecognizer.startRecognition(true);
        while (true) {

            String utterance = jsgfRecognizer.getResult().getHypothesis();

            if (utterance.startsWith("exit") || utterance.startsWith("stop") )
                break;
       }
```

5.1.6 Routing the Commands to Views

By keeping a map of commands to views we can swap layers of the NiftyDisplayPort screen. The router calls the HUDGUIState to open layers. The HUDGUIState class looks through the layers and loads the view called from the router.

```
// RouteList

List<Route> drivingSimRouteList = new ArrayList<Route>();

public DrivingSimulationRouteList(){

     drivingSimRouteList.add(new Route("DRIVER_HUD_VIEW", "DRIVERHUD", "HUDGUIController"));
}
```

```
//Router
            routeCommand(String command) {

                     this.hudgui.openView(simulation.getRoute(command));
            }
```

```
// HUDGUIState

for(Element view : screen.getLayerElements()){

               if (selectedview.getViewname().equals("DRIVER_HUD_VIEW") && !hudVisible)){

                       view.show();
}
```

5.1.7 Logging Commands with Slf4J

Logging can be important for auditing requirements. For example, perhaps, insight can be gained from logging which commands are used during certain driving situations.

Slf4J (Simple Logging Facade 4 Java) is a logging abstraction library that makes it easy to log multiple loggers via annotations:

```
@Log
   Creates private static final java.util.logging.Logger log =
   java.util.logging.Logger.getLogger(LogExample.class.getName());
@Log4j
   Creates private static final org.apache.log4j.Logger log =
   org.apache.log4j.Logger.getLogger(LogExample.class);
@Slf4j
   Creates private static final org.slf4j.Logger log =
   org.slf4j.LoggerFactory.getLogger(LogExample.class);
   ...


import lombok.extern.java.Log;
import lombok.extern.slf4j.Slf4j;

 @Log
 public class LogExample {

   public static void main(String... args) {
     log.error("Something's wrong here");
   }
 }

 @Slf4j
 public class LogExampleOther {

   public static void main(String... args) {
     if (utterance.equals("messages")) {
               log.info("sphinx recognised command: message");
       }
   }
 }
```

5.1.8 Simple Simulation with JavaFX

A simple simulation was implemented using a video of the driver's windscreen view of a car driving through streets of Sydney. This was achieved using a screen state with its own viewport, so that the HUD gui viewport could be rendered on top. This class utilised JavaFX for the video playback.

Figure 22.1 VoiceHud driver simulation showing main HUD view with speed and speed limit, as well as date view after date command.

Figure 22.2 VoiceHud driver simulation showing main HUD view with speed and speed limit, speed becomes red so driver immediately knows to slow down.

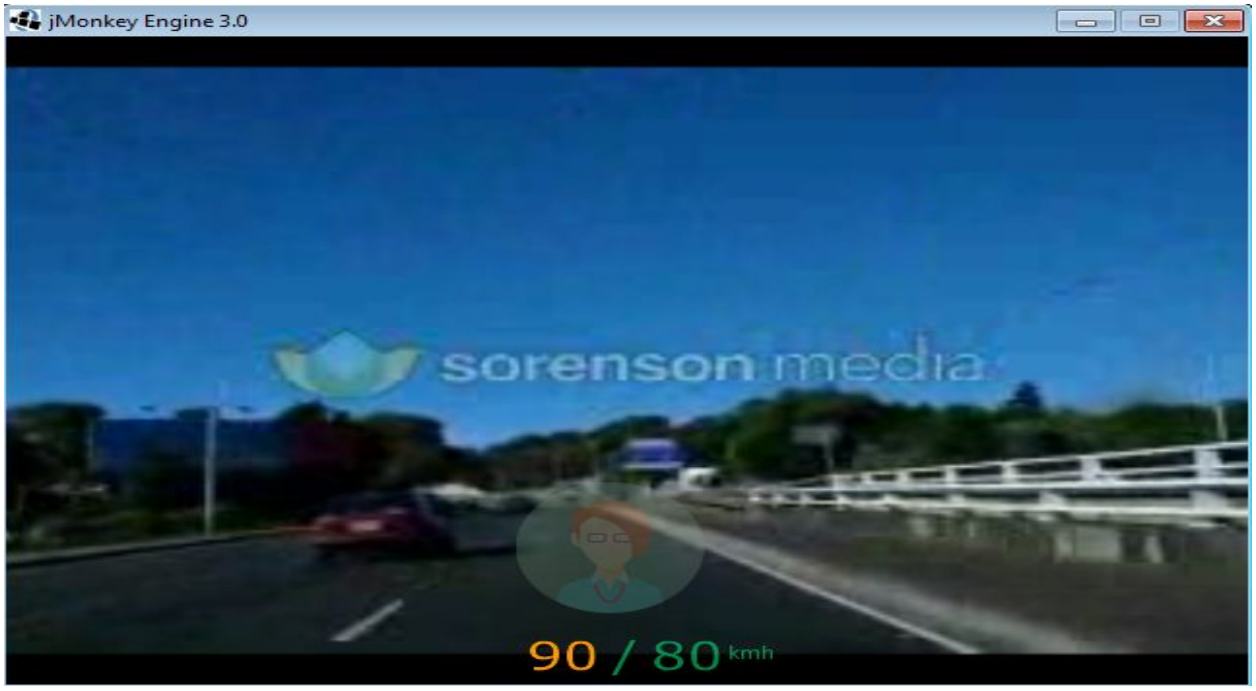


Figure 22.3 VoiceHud driver simulation showing main HUD view with speed orange when driver is near speed limit. Message view shows transparent photo briefly so driver identifies sender, followed by a readout of the message. Driver need not handle phone.

Figure 22.4 VoiceHud driver simulation showing main HUD view with map view showing briefly after the map command issued. Disappears after 2 seconds. Driver need not use HDD/GPS.

5.1.9 Advanced Simulation with openDS Integration

VoiceHud (driving) was partially integrated into OpenDS by setting up the main Simulator class to instantiate the core components used in VH (VoiceCommandManager, HUDGUIState, controllers and views) and setting the HUD to use the same Nifty viewport as openDS. Time limitations prevented a full integration as a self contained plugin.

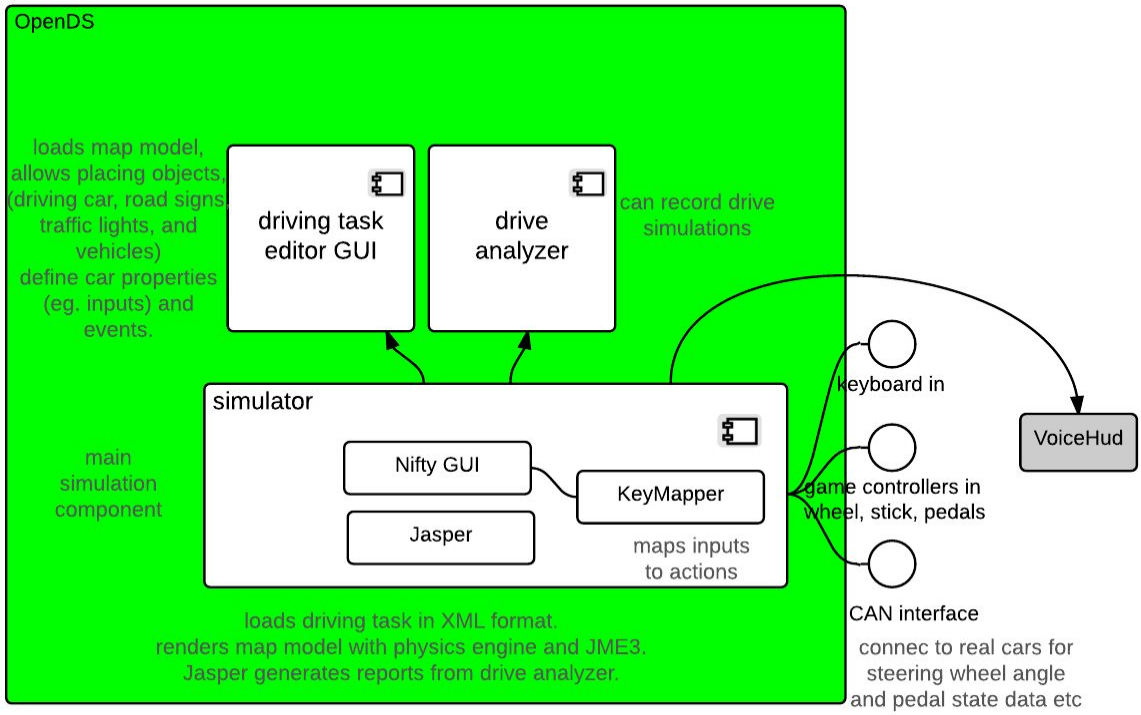
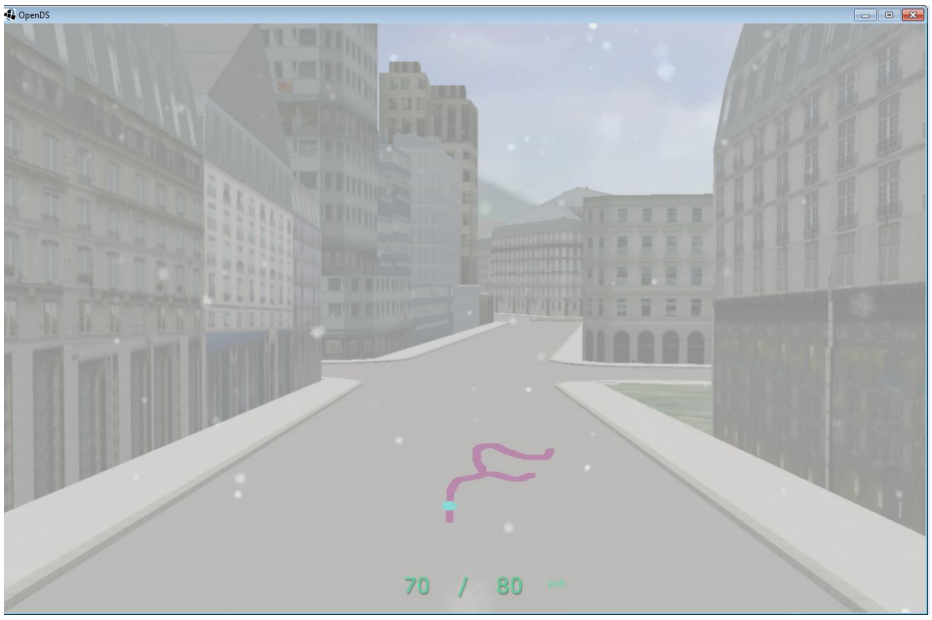


Figure 23. VoiceHud OpenDS integration Diagram

Figure 24. Simulator setup used for openDS integration and testing.
(left) Logitech G27 wheel, stick and pedals.
(middle) Shows the simulator running on windows desktop computer.
(right) Shows microphone headset Logitech G230 use with simulator.

5.1.9.1 Screenshots of VoiceHud in openDS

Figures 25.1 VH showing the date (also reads out date).



Figures 25.2 (top)
VH showing a message from mum (also read tts).
Both views showing the required speed limit and current speed.
Figure 25.3 (bottom)
VH showing the map view in a snowy condition simulation in openDS.

5.2 Evaluation

5.2.1 System Evaluation

After the system was developed and prior to user testing it was checked against the requirements and rated on how it achieved each. The table below shows all requirements met.

	Requirement	**achieved by**	**Achieved**
F1	The system must provide voice command recognition.	Core C2 voice	Yes
F2	Voice commands must be recognisable in real time.	Core C1 & C2	Yes
F3	The system must provide a HUD interface.	C3 hud	Yes
F4	The system must include text to speech capability for speech responses to the user.	C2	Yes
F5	The system must route voice commands to specific views of the HUD interface.	C4 router	Yes
F6	The system must provide functionality of activities normally done with other interfaces.	C3, V1-V8, plugins P1-P7	Yes
F7	The system must include a logger for auditing voice commands and other data.	Core C6	Yes

Table 6.1. System Testing Results

NF1	The HUD interface must only show a maximum of 3 view elements at once, so as not to incur too much cognitive load on the user.	C3 hides previous view on new view	Yes
NF2	The HUD interface must not be too distractive from the driving task.	C3 minimalist centric design	see UAT § 6.1
NF3	The HUD interface must not be too cluttered.	C3 few elements	Yes
NF4	Textual information on the HUD should be limited.	Particularly P1V4	Yes
NF5	Where possible information should be delivered to the user as speech.	V1-V4, V6-V8 P1, P3-P7	Yes
NF6	Where speech is not intuitive, the HUD should utilise visual information before textual information where this is intuitive.	V4 photo, V5 map. P1, P2	Yes
NF7	The system should be simple and easy to use so the user would use it over other interfaces.	easy commands C2, V1-V8	Yes
NF8	The voice command shouldn't chain commands deeper than 3 levels so as to prevent cognitive demand.	C2	Yes

Table 6.2 System Testing Results

5.2.2 System Issues

The two major issues with the system were concurrency and responsiveness.
The driving situation requires real time responsiveness of voice command recognition. But, the voice recogniser is used ‘off the shelf’ with only basic grammar specification. Consequently, the setup used in this project showed recognition of accuracy of only about 75%. There was also accuracy variation amongst users, for example, softer voices and accents that miss key phoneme articulations can be hard to recognise. Methods to optimise the recogniser include: modelling the acoustic environment, decreasing the sensitivity of the microphone listener, specification of a targeted language model and the use of a better quality microphone. Whilst these steps were deprioritised during prototype development they would be essential for deployment. For example, acoustic modelling would be essential for real deployment of the driving VH, given that the acoustics in a car are much different to the testing environment. Given that the grammar is small, these steps would go along way to mitigating these accuracy problems.

The system was also sometimes unresponsive, with some lag. This was despite testing on a computer with high memory and with ample memory dedicated to the java execution environment. To address real time recognition without delay, the CMU PocketSphinx implementation would be considered which is coded in C and a lot faster.

There are also concurrency issues that need further development to fix. For example, in the current implementation, the voice listening thread still listens whilst the text to speech thread speaks, and so the recognition tries recognising commands from this, which is incorrect and causes unexpected behavior from the user’s perspective.

5.2.3 Development Issues

The biggest developer issue was time management. Time resources were grossly overestimated in the context of other work commitments. Thus, not everything on the scope of the semester long project was completed. Due to context switching, this sometimes meant tools to help manage time such as trello were neglected somewhat. In contrast, the group project received a large amount of development time. Nevertheless, the use of sprints and buffers as described in section 4.1 mitigated these shortcomings. Another issue was catching up to speed on new third party frameworks with hefty but sometimes incomplete documentation.

6 RESULTS

6.1 System Performance Testing Results

Below are the results of the performance testing that was conducted prior to user acceptance testing. Each command was issued 20 times and the recognition rate (judged on whether the use case was achieved) was noted as well as scoring the command on lag out of 5. The absolute recognition was not as important as achieving the correct view. For example both “hud” and “hide” are linguistically very similar, so although hide achieved less recognition, and “hud” achieved more, they both still routed to the correct use case. That is, when the hud is shown, saying either of them should hide the HUD. When the HUD is not shown, the user would not use the command “hide” to show it and so the lower recognition ultimately still achieved the correct view. This testing was very simplistic and not robust to different voices and tones due to it being conducted only by the single tester, however it still derived a basic measure of system performance. The system scored 74% on average recognition of commands and 71% on average estimated lag.

Command	Use Case	Recognition Rate	Lag Noticed
hud	U1	0.85	4
hide	U9	0.6	4
fuel	U2	0.8	4
battery	U3	0.75	4
messages	U4	0.8	2
map	U5	0.75	4
music	U6	0.7	4
time	U7	0.7	3
date	U8	0.75	3
	Average	0.74	0.71

Table 7.1. System Performance Testing Results

6.2 Simple Simulation UAT Results

Five software engineering students were surveyed after testing the system (see Appendix. 9.7-9.8). They were then asked to score 10 questions on a scale of 1-5. Results are shown in the table below and a total score was calculated. As a simplistic indicator, a question's score was deemed positive if scored 3-5 on a positively phrased question or scored 1-2 on a negatively phrased question. The number of positive scores is noted in the right column. This gives a basic indicator of how well the system did for this question. The positive scores were averaged and multiplied by 20 to give a percentile total score of the system. A column listing which requirements the question supports is also shown.

q	**Questions**	**Req.**	Positive
1	I would use this system frequently	NF7	3
2	I found the system confusing	NF7	3
3	I thought the system was easy to use	NF7	3
4	I think the system would be frustrating to use in a real driving environment.	NF8	3
5	I found the commands in the system intuitive.	NF7	4
6	I thought the system was laggy / unresponsive / slow	F2	1
7	I think most people would learn to use this system easily and quickly	NF7	4
8	I think the interface was too distractive from the main driving task.	NF2	2
9	I think the system would keep my hands on the wheel more.	F1F2F5	3
10	I think the system would prevent mobile phone use during driving.	NF7	3
11	I think the system would keep my eyes on the road more.	F4	3
12	I dont think it would be accepted by older people.	NF7	2
13	I think the system would result in less accidents assuming it was completed, had an easy training process and had no latency.	NF2	3
14	I think the system would cause more accidents than without it.	NF2	3
Total Score			57.14%

Table 8.1. Simple VoiceHud Driver Simulation UAT Results

6.3 Advanced Simulation UAT Results

The same five software engineering students were surveyed after testing the advanced system. They were then asked 10 questions and to score on a scale of 1-5. The results are summarised in the table below and a total score was calculated. The results were quite similar, which was expected since the functionality and commands available didn't change, but the interface was slightly improved and the simulator was more realistic to the driving experience (even though the simple simulation was graphically more realistic). The results showed increased positive results on key questions (that map to crucial requirements).

q	**Questions**	**Req.**	Positive
1	I would use this system frequently	NF7	3
2	I found the system confusing	NF7	3
3	I thought the system was easy to use	NF7	3
4	I think the system would be frustrating to use in a real driving environment.	NF8	3
5	I found the commands in the system intuitive.	NF7	4
6	I thought the system was laggy / unresponsive / slow	F2	1
7	I think that most people would learn to use this system easily and quickly	NF7	4
8	I think the interface was too distractive from the main driving task.	NF2	3 (+1)
9	I think the system would keep my hands on the wheel more.	F1F2F5	4 (+1)
10	I think the system would prevent mobile phone use during driving.	NF7	4 (+1)
11	I think the system would keep my eyes on the road more.	F4	4 (+1)
12	I dont think it would be accepted by older people.	NF7	2
13	I think the system would result in less accidents assuming it was completed, had an easy training process and had no latency.	NF2	3
14	I think the system would cause more accidents than without it.	NF2	4 (+1)
Total Score			64.29%

Table 8.2 Advanced VoiceHud Driver Simulation UAT Results

7 GROUP PROJECT INTEGRATION

7.1 Brief Description of the Group Project

The final year software engineering class of 2015 at UON were tasked with designing and implementing a system that would improve the teaching of nursing and midwifery student tutorials. These tutorials involved case based learning with patient scenarios where the students had to progress through a clinical reasoning cycle (CRC) that involved: considering the patient's situation, collecting cues and information, processing this information and identifying problems and issues, establishing goals and outcomes for the patient, defining actions to achieve these goals, evaluating the effectiveness of these actions, and finally reflecting on the learning process. In particular, our task was to improve the learning process via improving student engagement within a collaborative group based setting using but not limited to multi-touch tables and incorporating ideas such as gamification, graphical visuals and touch based gestures. For reference, the full design document and final report for that system are provided **<here>.** Our group designed a system called Numitu (Nursing and Midwifery Tutorials) that provided activities mapping to the CRC including: a *Case Information* activity, an evidence gathering *Virtual Patient* activity, an *Issues and Evidence* activity where the evidence gathered is synthesized into issues, a *Goals and Actions* activity where goals were made to address the issues and actions to address the goals.

7.3.1 VoiceHud (Nursing) Use case Diagram

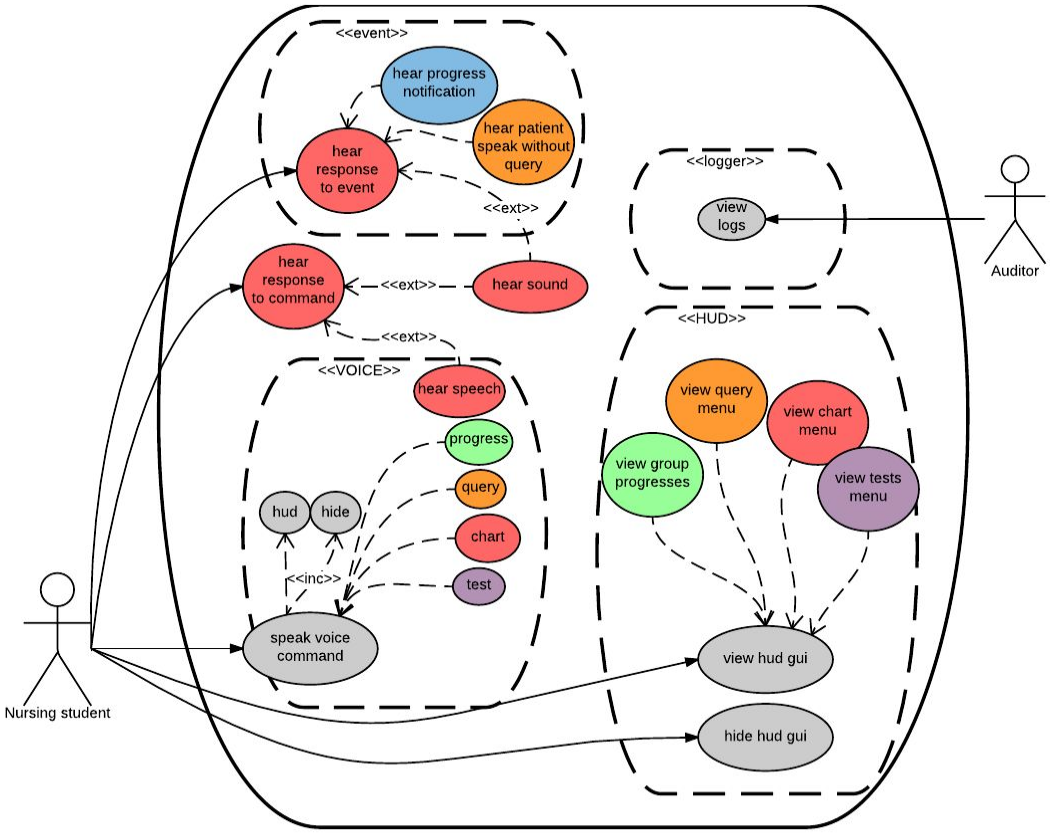


Figure 26. VoiceHud (Nursing Integration) Use Case Diagram

7.2 Initial Project Choice and Tool Choice

Initially, the group project inherited a java based touch supported architecture called Synergy Net [source] which also utilised Java Monkey Engine (JME) for graphics. It was in this context that i decided on my individual project using both Java and JME with the goal being to add voice commands with speech to text (STT) and responses with text to speech (TTS) to the Numitu system. As it turned out, Synergy Net wasn't well documented and was no longer supported by the original developers, and so we decided to rearchitecture the system from scratch using Javascript. This was some time into the development of both projects, as there was a significant period used for requirements gathering, project management and design. Thus, my individual project still used Java and JME and so to incorporate voice command and response into Numitu, i obviously decided to implement this using Javascript libraries. There were a few voice command libraries available, including *Annyang.js* , *Julius.js* and *pocketsphinx.js* (a port of CMU's pocketsphinx) and mainly one recent text to speech library called *meSpeak.js.* Annyang and meSpeak were chosen for their simplicity and stability.

7.3 Relevance to the Group Project

The group project is centered around teaching nursing students triage and the clinical reasoning cycle to ultimately make them better nurses. Communication is therefore a vital skill to develop, especially verbal communication when interacting with patients, as verbal communication is a key method of extrapolating information from patients as well as communicating across their care. This is again especially true with older patients, which was also a focus of the group project. However with non-voice based software systems, this key human element is lost. But with the group project using a virtual patient activity that simulates interaction with a real patient, the addition of voice command and dialog to this activity simulates this verbal human element. Furthermore, the group project was very focused around deployment to touchtables, the specific touch tables used having a relatively low resolution. Utilising a voicehud allows the system to move UI elements (such as the group progress timeline and menu button elements) off of the UI and onto a hidden HUD where it can be revealed by a voice command. Thus it achieves the utility of saving UI space when this is limited.

7.4 Speech To Text with Annyang.js

Whilst there were several javascript STT libraries available, the most matured and used one seemed to be Annyang.js. It was also relatively easy to setup. This was integrated into the group project with the module loader require.js which was a dependency of the group projects architecture.

7.5 Text To Speech with meSpeak.js

meSpeak.js (modularly enhanced speak.js) was the most supported and updated version of speak.js which was the main TTS library used by developers. Again, this library was integrated easily using require.js

7.6 Implementation and Integration

The implementation involved installing the npm package manager versions of annyang and meSpeak, including it in the bower configuration that the group project used and requiring it in the virtual patient component that demonstrated the integration (on an isolated branch). It then involved defining commands that the recogniser library would recognise and actions (method calls) based on those commands. One caveat of using Annyang and other libraries that utilise the web speech API (for example when using the chrome browser), the library does not remember the decision to allow the microphone to listen. Therefore, SSL over https protocol is required to remember this. To implement this without hosting, a self signed certificate and key was used.

```
var Annyang = require('annyang');

if(annyang){
      if command = "query"{
            query.button.show();
      }

      if command = "howareyou"{
            tts.respond("not very well");
      }
      if command = "hud"{
           hud.show();
      }
}
```

Code Fragment Showing the use of Annyang.js to implement a VoiceHud for Nursing students.

Above, the javascript library is required as an object using require.js.
The voice listener then recognises commands and the respective logic is executed.
Here, the method calls that bring up the menus are executed instead of onclick/touch event.
The hidden HUD is also revealed using the hud command.

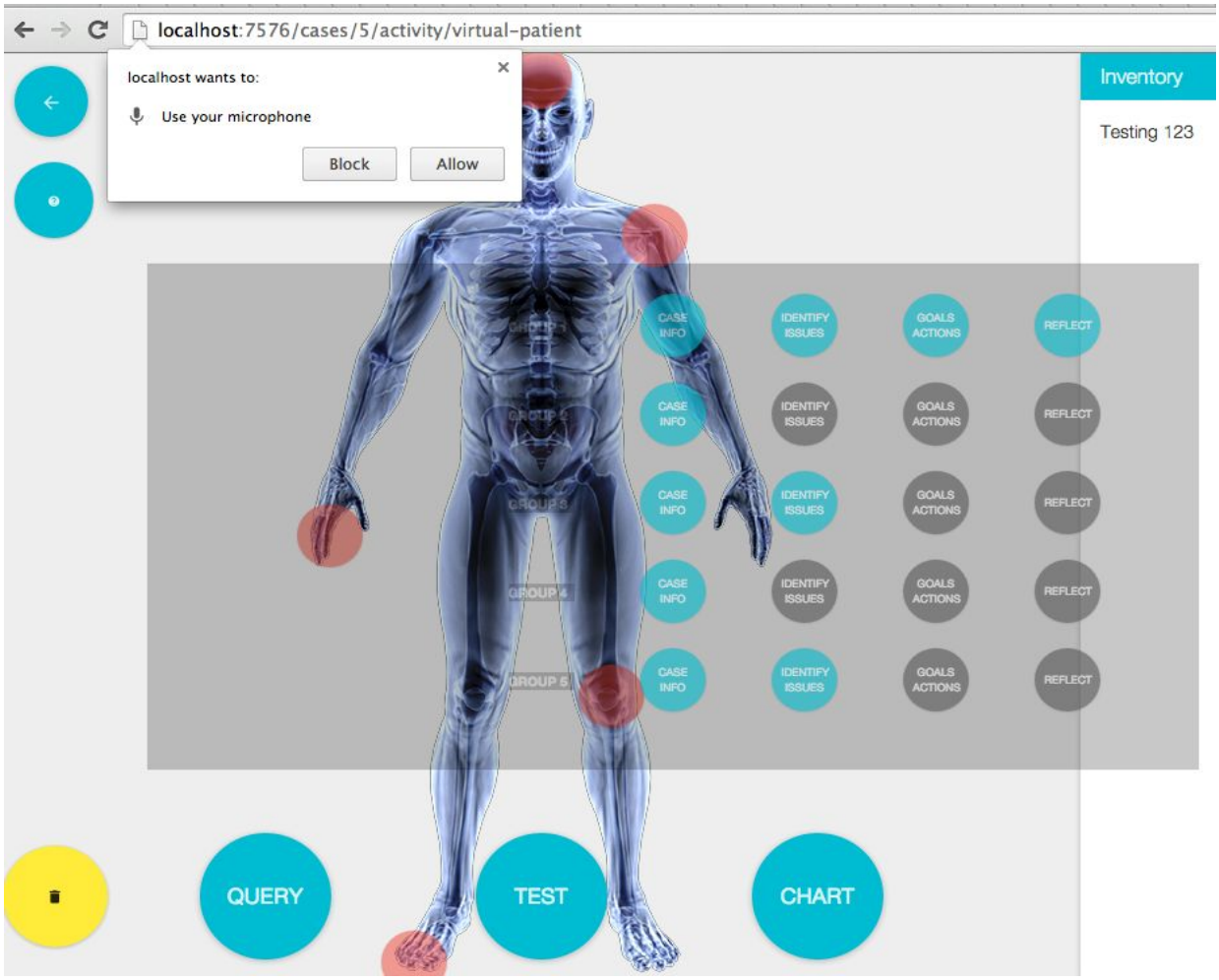


Figure 27.0 Screenshot of VoiceHud integration in Numito nursing collaborative application. The application asks the user if it can listen using their microphone. If the user allows, then it listens for commands (continually over https). The student can then issue "progress" command to view the team progress through the tutorial activities. The student can also bring up the menu's that are also here accessible by touch and click but could also be hidden to regain more UI space.

7.7 Numito Integration System Performance Testing Results

System performance testing was conducted before user acceptance testing in the same manner as in section 6.1. The integrated system scored 67% on average recognition of commands and 56% on average estimated lag.

Command	Recognition Rate	Lag Noticed
hud	0.75	3
hide	0.6	3
query	0.6	2
test	0.7	3
chart	0.7	3
Average	0.67	0.56

Table 9.0. System Performance Testing Results For Numito Integration

7.8 VoiceHud – Numito Implementation UAT

The same five software engineering students were also surveyed after using the integration of voicehud in Numito as per Appendix 9.9, again using a modified SUS test. The results are summarised in the table below and a total score is calculated in the same manner as § 6.2-6.3.

The system scored lower than the two driving simulations, which was expected since the integration was not as completely implemented and also the Annyang library used seemed less accurate at recognising words and robust to different voices and tones than did Sphinx4.
As well as this, the text to speech library used was very robotic and was probably off putting to the goal of mimicking a real patient. This is evident in question 4's low positive rate.

q	**Questions**	Positive
1	If I were a nursing student, I would use this feature.	4
2	I found the feature unresponsive	3
3	I thought the feature made the activity more realistic	4
4	I think the reply voices were too unnatural	2
5	I thought the voicehud was good for saving interface space	3
6	I thought the voicehud was laggy.	2
7	I think that the voice command complements the touch based interaction.	4
8	I think the voice commands would be tricky to use in group tutorials.	4
9	I think the query feature in the virtual patient was more realistic due to the voice interaction.	3
10	I would prefer to use touch interaction only.	3
Total Score		60%

Table 10. VoiceHud Numito (FYP Collaborative Nursing Tutorials) Integration UAT Results

8 CONCLUSION

Arguably the two most important human modes of interaction are voice and vision. Human speech is an essential part of human communication and without vision we are in the dark and cannot perform many high level activities like driving a car. We have evolved to utilise our voice and vision concurrently whilst controlling our arms and legs yet voice is underutilised as a human to computer interface controller in the driving situation, and the research literature shows that a well designed voice interface can be less distracting than using touch based interfaces that both distract vision and focus away from the useful field of view and require the driver to take at least one hand off the wheel. The research also shows that HUD interfaces are much less distracting than HDD interfaces.

With increased usage of mobile phones, and continual usage of HDDs it is important that the VoiceHud concept be fully tested since it may prevent accidents and potentially save lives. Therefore, the VoiceHud project aimed to design, implement and test a prototype voice commanded HUD interface. Whilst the focus was on the driving situation, the java based system was designed in an extensible manner such that it could be used in other domains and for other simulations.

Good software design principles were used to develop a prototype system that was tested on performance of command recognition with around 75% accuracy and minimal lag only noticed on one command due to competing processing. The system was then user tested on two simulations, a basic one utilising a movie playback of a car windscreen view driving through busy streets, and an advanced one that integrated the VoiceHud into a cheap but high quality open source driving simulator. These user acceptance test results were largely positive scoring 57% and 64% respectively. The scores were particularly strong on questions targeting the key project aims of keeping the driver's eyes on the road, hands on the wheel and preventing phone use. Users also agreed the system was easy to use and intuitive. The users did score poorly on older people accepting the system and the question of whether the system would be less or more distractive was on the whole inconclusive. Despite, these mostly positive results, the sample size was very small (5), made up of only software students from the same working group (FYP) who were all under the age of 30, so the results were ultimately unrepresentative, potentially biased, and inconclusive but due to project time and resource limitations were better than none.

The VoiceHud concept, however, was indeed demonstrated to be transferrable to the domain of nursing with the integration of a VoiceHud javascript implementation facilitating the display of group progress and accessing menus in the group learning situation. Not only did this integration bring a natural human component to the human to computer interaction that was the virtual patient activity, but it provided access to a progress screen without taking up more real estate with another GUI reveal buttons.

The nursing integration was also tested using the same performance method as with the driver voicehud this time scoring lower on both recognition rate (67%) and lag (56%). The system was then tested using the same five users and the results scored slightly worse (60%) than in the driving simulation. This was possibly due to the less accuracy or recall of the Annyang voice recognition library, as well as potentially due to users seeing it integrated less completely, as well as a more robotic and unnatural voice used by the meSpeak text to speech responses. Despite this, the users generally agreed that the concept leant itself to interfaces with limited real estate and virtual human interactions (which perhaps could also be utilised more interacting with AI characters in gaming). On the other hand, some users expressed the view that the concept is somewhat counterintuitive to the collaborative environment, since it was seen to assume one person commanding the interface. On one hand the problem of control is a valid concern and the demonstrated prototype doesn't discern between who is issuing commands, on the other hand it could be argued this in itself is providing opportunity for users to collaborate in the issuing of commands. The problem of discerning control of users could be solved by the combination of more accurate recognition with reliable voice identification. This would present the opportunity for the system to be a more intelligently collaborative participant and interpreter of commanders and of the group as a whole, and better mimic a real life person. The problem of control indeed needs to be considered in the driving environment too when discerning commands from driver and passenger.

The limitations of the implementation of the driver voicehud include the unutilised events system to update the GUI when crucial events (like speeding) occur. A plugin that receives real data from the car (such as from the OBD 2 port) could realise this. Alternatively, the simulation could be extended to mock events. The system is also not optimised for some competing processing such as voice recognition whilst the text to speech component speaks text. This could easily be fixed by pausing the recognition during speak back, but time constraints of the project did not allow this. The limitations of the integration with the group FYP include a lower recognition rate and a very limited set of implemented commands, due to this mainly being a "throwaway prototype" whilst the driver project was the main focus as a prototype to keep and iterate on.

In conclusion, whist the project didn't achieve the level of completion nor reliability in user testing that it desired, predominantly due to time constraints and limited access to testers, it still achieved revealing preliminary results on the design and implementation of a VoiceHud for the driving and nursing situations as well as achieving a good framework for implementations of VoiceHud's as well as those that need to use a simulation. The project succeeded in providing a solid platform for improving and iterating the VoiceHud concept on.

Future work should continue testing the VoiceHud concept in the driving situation and other situations where distraction and cluttered or real estate restricted interfaces are involved.

9 APPENDICES

9.1 Driver Focus Areas

Figure 28. Focus areas highlighted in orange and red, described by green text. Red text shows distractive heads down interfaces.

Figure 29. Useful field of view (UFOV) decrease with age [67]

9.2 Driving distraction

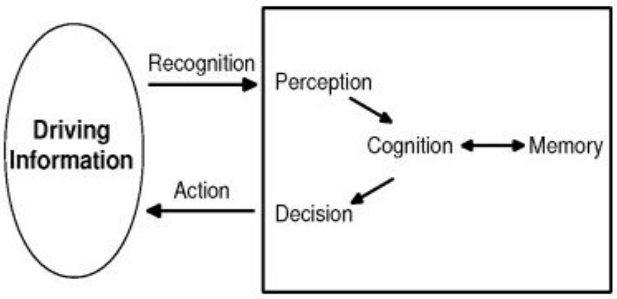
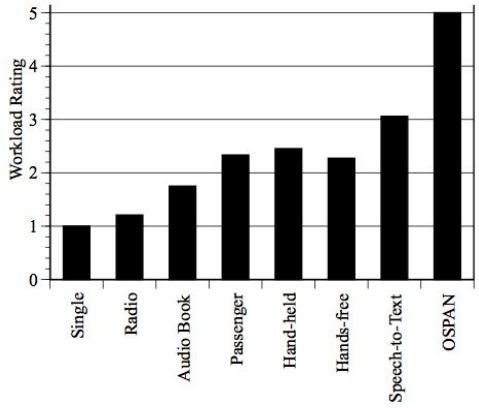


Figure. 30 A model of driver information processing.



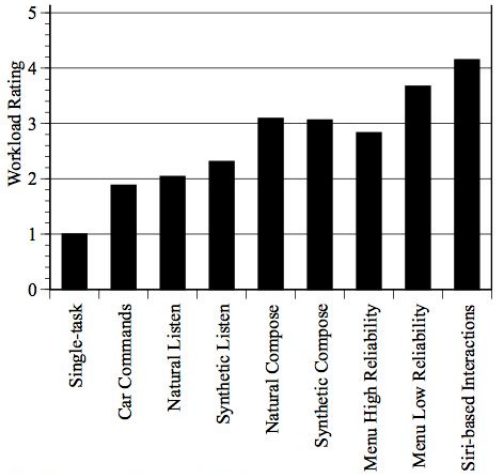


Figure. 31.1 (Left) Workload Rating Scale, Strayer (2013)[34].
Figure. 31.2 (Right) Expanded Wordload Rating Scale, Strayer (2014)[35].

9.3 Examples of HDD's

Figure 32.1 (left) 2011 Chevrolet Volt Dash and Central Display Unit [68]
Figure 32.2 (middle) 2014 Tesla Model X Dash and Central Display Unit [69]
Figure 32.3 (right) 2015 Apple CarPlay Central Display Unit [70]

9.4 Examples of HUD's

Fig. 33 Aircraft HUDs: (left) Boeing 747 HUD [71] (right) FA18 HUD [72]

Fig 34. First commercial HUD: (left) Manual of Oldsmobile HUD [73] (right) photo of in use [74]

Fig 35. Some early HUDs
(left) BMW E60 HUD [75] (right) 2013 Toyota Prius HUD [76]

Fig. 36 Pioneer NavGate [77]

Fig. 37 Sygic Mobile Application HUD [78]

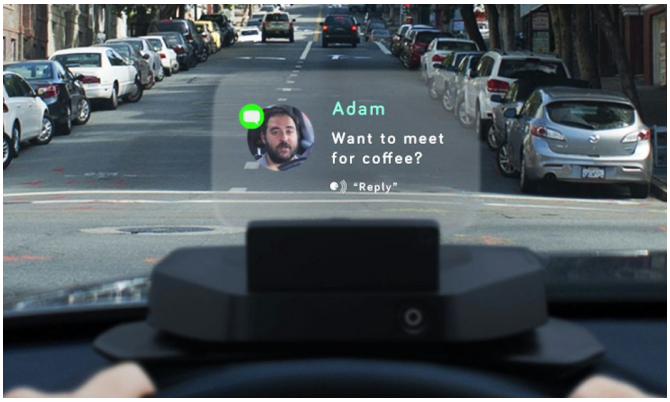


Fig. 38 Navdy HUD [79]

9.5 Design Guidelines and Recommendations

Excerpts from HARDIE Guidelines
Guidance should be presented in the form of simple, step-by-step instructions at the time of each driving operation. Drivers cannot be expected to process complex information when determining their desired route. In other words, the system may not display a map with a highlighted route.
Road layout information should be presented in abbreviated form.
Text should be avoided on maps whenever possible. If text overlays are necessary, they should be oriented horizontally.
If map information is required while driving, the direction of forward movement should be at the top.
Sound messages should be as simple as possible, limited ideally to the direction of the driving operation.
The information presented in a route guidance system must match the information actually on the roads.

Table 11. HARDIE Guidelines **[46]**

System	*Guidelines*
IVIS	Use simple information.
	Use multi-modal displays for complex and critical information.
	Control the use of auditory components to avoid annoyance.
	Present textual information by vocal announcement.
	Limit the number of guidance instructions to be read to no more than two.
	Use symbols to present guidance information.
	Task time should not be more than 20 seconds.
	Glances to display should not be longer than 2.5 seconds.
	Tasks should not affect lateral and longitudinal vehicle control.
	Tasks should not affect driver's workload and situational awareness.
	System feedback should be instantaneous.
	Avoid flashing or moving graphics elements.
	Minimize visual clutter.
	Use maximum contrast between display elements.
	Consider color blindness and use colors sparingly.
	Keep backgrounds simple and muted.
	Group information logically considering frequency and sequence of use.
	Consider user's behavior and needs.
	Let user set pace and initiate interaction.
	Prioritize information.
	Accommodate for experience.
	Restrict information when necessary.
	Auditory icons should use easily recognizable sounds.
	Auditory icons should not sound similar.
	Auditory icons should have similar duration, intensity, and quality.
VISUAL DISPLAYS*	Color-luminance: Red at 50 cd/m^2.
	Temporal characteristics: Intermittent cycle = 0.2 sec, Rate = 30%
	Spatial characteristics: Visual angle = 6.0 deg, Shape = contoured square
AUDITORY DISPLAYS*	Sound pressure level = 80 dBA
	Frequency: Fundamental tone = 2.0 kHz, Secondary = 1.5 x fundamental
	Temporal characteristics: Intermittent cycle = 0.2 sec, rate = 70%

Table 12. Human Factors recommended guidelines for in-vehicle displays [47]

9.6 Lct

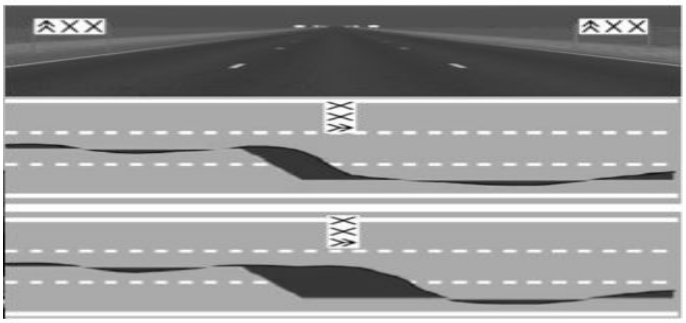


Fig. 39.1 Original LCT showing road (top) driver performance (bottom).[77]

Fig. 39.2 Original LCT Test Color [81]

Fig. 39.3 LCT Test as in OpenDS Simulator [81]

9.7 UAT Consent Forms

User Acceptance Testing Consent Form

V.1.0 Date 27.10.2015

VoiceHud - Basic Driving Simulation Demo

A voice commanded HUD interface for safer driving.

VoiceHud is a user interface that allows you to use voice command to access information and graphics views via a Heads Up Display (HUD). During this test you will be the user in a basic driving simulation. This involves a video playback of the front windscreen view of a car driving through busy city roads. You will be asked to imagine you are driving the car and to pretend you are driving the car with the provided steering wheel, pedals and gearstick. You will be told once the simulation is setup and ready to start. You will be asked to speak commands to the system whilst pretending to drive and to observe the HUD displayed. Finally you will be asked to complete the survey form provided.

Please read and sign the consent form below before partaking in the simulation.

I agree to participate in the above research project and give my consent freely.

I understand that the project will be conducted as described in the Information Statement, a copy of which I have retained.

I understand I can withdraw from the project at any time and do not have to give any reason for withdrawing.

I consent to

- Test the described software system
- Complete the accompanied questionnaire.

I understand that my personal information will remain confidential to the researchers.

I have had the opportunity to have questions answered to my satisfaction.

Print Name: _____

Signature: _____ **Date:** _____

User Acceptance Testing Consent Form

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

V.1.0 Date 27.10.2015

VoiceHud – Advanced openDS Driving Simulation Demo

A voice commanded HUD interface for safer driving.

VoiceHud is a user interface that allows you to use voice command to access information and graphics views via a Heads Up Display (HUD). During this test you will be the user in a basic driving simulation. This involves driving using a driving simulator called openDS. You will be asked to drive a simulated car using the provided steering wheel, pedals and gearstick. You will be told once the simulation is setup and ready to start. You will be asked to speak commands to the system whilst driving and to observe the HUD displayed. You will then be asked to complete a simulated lane change test both without the voicehud and utilising the voicehud. Finally you will be asked to complete the survey form provided.

Please read and sign the consent form below before partaking in the simulation.

I agree to participate in the above research project and give my consent freely.

I understand that the project will be conducted as described in the Information Statement, a copy of which I have retained.

I understand I can withdraw from the project at any time and do not have to give any reason for withdrawing.

I consent to

- Test the described software system
- Complete the accompanied questionnaire.

I understand that my personal information will remain confidential to the researchers.

I have had the opportunity to have questions answered to my satisfaction.

Print Name: _____

Signature: _____ **Date:** _____

9.8 UAT Questions VoiceHud

VoiceHud - Basic Driving Simulation UAT

	Questions	Strongly Disagree				Strongly Agree
		1	2	3	4	5
1	I would use this system frequently					
2	I found the system confusing					
3	I thought the system was easy to use					
4	I think the system would be frustrating to use in a real driving environment.					
5	I found the commands in the system intuitive					
6	I think the system was too laggy / unresponsive / slow.					
7	I think that most people could learn to use this system quite easily and quickly					
8	I think the interface was not distractive from the main driving task.					
9	I think the system would keep my hands on the wheel more.					
10	I think the system would prevent mobile phone use during driving.					
11	I think the system would keep my eyes on the road more.					
12	I think the system would result in less accidents assuming it was completed, had an easy training process and had no latency.					

VoiceHud – Advanced openDS Driving Simulation UAT

	Questions	Strongly Disagree				Strongly Agree
		1	2	3	4	5
1	I would use this system frequently					
2	I found the system confusing					
3	I thought the system was easy to use					
4	I think the system would be frustrating to use in a real driving environment.					
5	I found the commands in the system intuitive					
6	I think the system was too laggy / unresponsive / slow.					
7	I think that most people could learn to use this system quite easily and quickly					
8	I think the interface was not distractive from the main driving task.					
9	I think the system would keep my hands on the wheel more.					
10	I think the system would prevent mobile phone use during driving.					
11	I think the system would keep my eyes on the road more.					
12	I think the system would result in less accidents assuming it was completed, had an easy training process and had no latency.					

Table 13. Copy of the Original UAT form shown.

Two additional questions were added (written on) both surveys :

I think the system would cause more accidents than without it.
I dont think it would be accepted by older people.

9.9 UAT Numitu Integration Questions

	Questions	Strongly Disagree				Strongly Agree
		1	2	3	4	5
1	If I were a nursing student, I would use this feature.					
2	I found the feature unresponsive					
3	I thought the feature made the activity more realistic					
4	I think the reply voices were too unnatural					
5	I thought the voicehud was good for saving interface space					
6	I thought the voicehud was laggy.					
7	I think that the voice command complements the touch based interaction.					
8	I think the voice commands would be tricky to use in group tutorials.					
9	I think the query feature in the virtual patient was more realistic due to the voice interaction.					
10	I would prefer to use touch interaction only.					

Table 14. Simple VoiceHud Driver Simulation UAT Results

9.10 User Guide

A copy of the user guide is available on github at https://github.com/jskye/voicehud/tree/master/docs but is shown below:

VoiceHud Setup Guide

1. Clone the project `git clone https://github.com/jskye/voicehud.git`
2. Ensure that Maven is installed.
3. Import the maven project (using the pom.xml file) into Eclipse (tested) or some other IDE.
4. Download one of the two video files below required for the simulation
   https://www.dropbox.com/s/ni7ufh01etvnoud/sydriveHQ.flv?dl=0 (bigger)
   https://www.dropbox.com/s/g7a8q7lg7ss1hbu/lct1HQ.flv?dl=0 (smaller)
1. Move the video you downloaded into the Assets/Video/ folder of the project.
2. Ensure the following line in `SimpleMoviePlayer.java` refers to that video file:

```
media = newMedia(newFile(workingDir+"/assets/Video/sydriveHQ.flv").toURI().toASCIIString());
```

   By default its set to use the SydriveHG file, as this was used in UAT.

1. Ensure the project execution environment is set to java 1.8
   (project > properties > Java Compiler >Compiler Compliance Level)
2. Run `mvn install`
3. Maven shouldve download src's from the dependencies.
   If it didnt, run `mvn eclipse:clean`
   it will clean the project as well as download dependency jars
4. Run the App.java by selecting the class and running as java app or maven configuration:
   (a) Right click > Run As > Java Application
   (b) Run > Run As > Maven Build or `mvn:build`
5. To run tests, run Maven Test configuration or `mvn:test`

VoiceHud User Guide

1. Run the App.java as in the setup.
2. The JME console will come up.
   Choose display settings and click continue.
3. A black screen will load. This is for a loading screen that's not yet implemented.
4. Wait until the video loads.
5. Once the video is playing back (note. the command line output should say that the voicehud is listening), you can now start issuing commands.
6. Make sure you speak with a decent amount of volume into your microphone and that your microphone is turned up in your O.S device settings.
7. Command `HUD` brings up the HUD interface.
8. Try out these other commands:
   a. Time
   b. Date
   c. Fuel
   d. Messages
   e. Map

10 REFERENCES

[1] Stutts, Jane et al. "Driver's exposure to distractions in their natural driving environment." *Accident Analysis & Prevention* 37.6 (2005): 1093-1101.

[2] Fischer, Edith, and Richard F Haines. "Cognitive issues in head-up displays." (1980).

[3] Wickens, Christopher D et al. "Cognitive factors in aviation display design." Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE 1998: E32/1-E32/8 vol. 1.

[4] Kiefer, RJ. "A review of driver performance with head-up displays." Intelligent Transportation: Realizing the Future. Abstracts of the Third World Congress on Intelligent Transport Systems 1996.

[5] Regan, Michael A, Charlene Hallett, and Craig P Gordon. "Driver distraction and driver inattention: Definition, relationship and taxonomy." *Accident Analysis & Prevention* 43.5 (2011): 1771-1781.

[6] <http://www.ors.wa.gov.au/Campaigns-Programs/Driver-Distraction> accessed June 2015.

[7] Kircher, Katja. "Driver distraction: a review of the literature." (2007).

[8] National Highway Traffic Safety Administration. "NHTSA.(2012)." *Visual-Manual NHTSA Driver Distraction Guidelines for In-Vehicle Electronic Devices*.

[9] Poole, Alex, and Linden J Ball. "Eye tracking in HCI and usability research."

[10] Simons, Daniel J. "Attentional capture and inattentional blindness." *Trends in cognitive sciences* 4.4 (2000): 147-155.

[11] Strayer, David L, and Frank A Drews. "Cell-phone–induced driver distraction." Current Directions in Psychological Science 16.3 (2007): 128-131.

[12] Hyman, Ira E et al. "Did you see the unicycling clown? Inattentional blindness while walking and talking on a cell phone." *Applied Cognitive Psychology* 24.5 (2010): 597-607.

[13] Mack, Arien, and Irvin Rock. *Inattentional blindness*. Cambridge, MA: MIT press, 1998.

[14] Ünal, Ayça Berfu, Linda Steg, and Kai Epstude. "The influence of music on mental effort and driving performance." Accident Analysis & Prevention 48 (2012): 271-278.

[15] Brodsky, Warren, and Zack Slor. "Background music as a risk factor for distraction among young-novice drivers." *Accident Analysis & Prevention* 59 (2013): 382-393.

[16] Ünal, Ayça Berfu, Linda Steg, and Kai Epstude. "The influence of music on mental effort and driving performance." *Accident Analysis & Prevention* 48 (2012): 271-278.

[17] http://www.topnews.in/files/car_radio.jpg

[18] van der Zwaag, Marjolein D et al. "The influence of music on mood and performance while driving." *Ergonomics* 55.1 (2012): 12-22.

[19] Jensen, Brit Susan, Mikael B Skov, and Nissanthen Thiruravichandran. "Studying driver attention and behaviour for three configurations of GPS navigation in real traffic driving." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* 10 Apr. 2010: 1271-1280.

[20] Ross, Tracy et al. "HARDIE Design Guidelines Handbook." Human Factors Guidelines for Information Presentation by ATT Systems 530 (1996).

[21] Hajime, ITO et al. "VISUAL DISTRACTION WHILE DRIVING: Trends in Research and Standardization." IATSS Research 25.2 (2001): 20-28.

[22] Fumero-AguilÃ, MarÃa C. "Development of Guidelines for In-Vehicle Information Presentation: Text vs. Speech." (2004).

[23] Strayer, David L, Frank A Drews, and William A Johnston. "Cell phone-induced failures of visual attention during simulated driving." Journal of experimental psychology: Applied 9.1 (2003): 23.

[24] Strayer, David L, and Frank A Drews. "Cell-phone–induced driver distraction." Current Directions in Psychological Science 16.3 (2007): 128-131.

[25] Cuřín, Jan et al. "Dictating and editing short texts while driving: Distraction and task completion." Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications 30 Nov. 2011: 13-20.

[26] Liu, Yung-Ching. "Effects of using head-up display in automobile context on attention demand and driving performance."Displays 24.4 (2003): 157-165.
[27] Jæger, Mads Gregers, Mikael B Skov, and Nils Gram Thomassen. "You can touch, but you can't look: interacting with in-vehicle systems." Proceedings of the SIGCHI Conference on Human Factors in Computing Systems 6 Apr. 2008: 1139-1148.
[28] Kiefer, Raymond J. "Effect of a head-up versus head-down digital speedometer on visual sampling behavior and speed control performance during daytime automobile driving." 1 Feb. 1991.
[29] Delogu, Cristina, Stella Conte, and Ciro Sementina. "Cognitive factors in the evaluation of synthetic speech." Speech Communication 24.2 (1998): 153-168.
[30] Harbluk, Joanne L, and Simone Lalande. "Performing e-mail tasks while driving: The impact of speech-based tasks on visual detection." Proceedings of the 3rd International Driving Symposium on Human Factors in Driving Assessment, Training and Vehicle Design 27 Jun. 2005: 304-310.
[31] Nunes, Luis, and Miguel Angel Recarte. "Cognitive demands of hands-free-phone conversation while driving." Transportation Research Part F: Traffic Psychology and Behaviour 5.2 (2002): 133-144.
[32] Vollrath, Mark, Dipl-Psych Jannette Maciej, and Dipl-Psych Ute Niederée. "In-car distraction study final report."Technical University of Braunschweig, Braunschweig (2008).
[33] Zheng, Pengjun, Mike McDonald, and Carl Pickering. "Effects of intuitive voice interfaces on driving and in-vehicle task performance." Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on 12 Oct. 2008: 610-615.
[34] Strayer, David L et al. "Measuring cognitive distraction in the automobile." (2013).
[35] Cooper, Joel M, Hailey Ingebretsen, and David L Strayer. "Mental Workload of Common Voice-Based Vehicle Interactions across Six Different Vehicle Systems." (2014).
[36] Previc, Fred H. Spatial disorientation in aviation. Aiaa, 2004: 174
[37] Sojourner, Russell J, and Jonathan F Antin. "The effects of a simulated head-up display speedometer on perceptual task performance." Human Factors: The Journal of the Human Factors and Ergonomics Society 32.3 (1990): 329-339.
[38] Burnett, Gary E. "A road-based evaluation of a head-up display for presenting navigation information." Proceedings of the HCI international conference, Crete 23 Jun. 2003.
[39] Charissis, Vassilis, Stylianos Papanastasiou, and George Vlachos. "Comparative study of prototype automotive HUD vs. HDD: collision avoidance simulation and results." 14 Apr. 2008.
[40] Liu, Yung-Ching, and Ming-Hui Wen. "Comparison of head-up display (HUD) vs. head-down display (HDD): driving performance of commercial vehicle operators in Taiwan." International Journal of Human-Computer Studies 61.5 (2004): 679-697.
[41] Weinberg, Garrett, Bret Harsham, and Zeljko Medenica. "Evaluating the usability of a head-up display for selection from choice lists in cars." Proceedings of the 3rd International Conference on Automotive User Interfaces and Interactive Vehicular Applications 30 Nov. 2011: 39-46.
[42] Bengler, Ablaßmeier, Markus et al. "Eye gaze studies comparing head-up and head-down displays in vehicles." *Multimedia and Expo, 2007 IEEE International Conference on* 2 Jul. 2007: 2250-2252.
[43] Miura, Toshiaki. "Active function of eye movement and useful field of view in a realistic setting." (1990).
[44] Ball, Kathleen, and C Owsley. "The useful field of view test: a new technique for evaluating age-related declines in visual function." Journal of the American Optometric Association 64.1 (1993): 71-79.
[45] Aguiló, María C Fumero. "Development of guidelines for in-vehicle information presentation: Text vs. speech." 12 May. 2004.

[46] Campbell, John Lyle, C Carney, and Barry H Kantowitz. "Human factors design guidelines for advanced traveler information systems (ATIS) and commercial vehicle operations (CVO)." 1998.
[47] Ross, Tracy et al. "HARDIE Design Guidelines Handbook." Human Factors Guidelines for Information Presentation by ATT Systems 530 (1996).
[48] Watanabe, Hiroshi et al. "The effect of HUD warning location on driver responses." Sixth World congress on Intelligent Transport Systems, Toronto, Canada Nov. 1999.
[49] Tsimhoni, Omer, Paul Green, and Hiroshi Watanabe. "Detecting and reading text on HUDs: effects of driving workload and message location." 11th Annual ITS America Meeting, Miami, FL, CD-ROM2001.
[50] Huang, Cheng-Hung et al. "The Effects of Interface Design for Head-Up Display on Driver Behavior." Life Science Journal10.2 (2013): 2058-2065.
[51] Kiefer, Raymond J. "Quantifying head-up display (HUD) pedestrian detection benefits for older drivers." 16th International Technical Conference on Experimental Safety Vehicles. Windsor: NHTSA Oct. 1998: 428-437.
[52] <http://imaginyze.com/Site/Screenshots_files/Media/IMG_6462/IMG_6462.jpg> Accessed 8 June 2015.
[53] Rumar, Kåre. "Night vision enhancement systems: What should they do and what more do we need to know?." (2002).
[54] Murer, Martin et al. "Exploring the back of the steering wheel: Text input with hands on the wheel and eyes on the road." Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Applications 17 Oct. 2012: 117-120.
[55] Bach, Kenneth M et al. "Evaluating driver attention and driving behaviour: comparing controlled driving and simulated driving." Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1 1 Sep. 2008: 193-201.
[56] <http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/eye-tracking-hmd-upgrade.html> Accessed 8 June 2015.
[57] Mattes, Stefan. "The lane-change-task as a tool for driver distraction evaluation." Quality of work and products in enterprises of the future (2003): 57-60.
[58] Mattes, Stefan, and Anders Hallén. "Surrogate distraction measurement techniques: The lane change test." Driver distraction: Theory, effects, and mitigation (2009): 107-121.
[59] Harbluk, Joanne L et al. "Using the lane-change test (LCT) to assess distraction: Tests of visual-manual and speech-based operation of navigation system interfaces." Proceedings of the 4th international driving symposium on human factors in driver assessment, training, and vehicle design 9 Jul. 2007: 16-22.
[60] "ISO 26022:2010 - Road vehicles -- Ergonomic aspects of ..." 2009. 8 Jun. 2015 <http://www.iso.org/iso/catalogue_detail.htm?csnumber=43361>
[61] Harbluk, Joanne L et al. "Using the lane-change test (LCT) to assess distraction: Tests of visual-manual and speech-based operation of navigation system interfaces."Proceedings of the 4th international driving symposium on human factors in driver assessment, training, and vehicle design 9 Jul. 2007: 16-22.
[62] Hofmann, Peter, Gerhard Rinkenauer, and Dietmar Gude. "Head-up-displays support response preparation in a lane change task." Proceedings of the Human Factors and Ergonomics Society Annual Meeting 1 Sep. 2008: 1233-1237.
[63] <http://www.opends.de> Accessed June 2015.
[64] Weinberg, Garrett, and Bret Harsham. "Developing a low-cost driving simulator for the evaluation of in-vehicle technologies." Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications 21 Sep. 2009: 51-54.

[65] L.Kai-Fu, H.Hsiao-Wuen, R. Raj(1990) An Overview of the Sphinx Speech Recognition System. IEEE Transection Of Acoustic Speech and SignalProcessing,Vol38. <http://www.ri.cmu.edu/pub_files/pub2/lee_k_f_1990_1/lee_k_f_1990_1.pdf>
[66] Huggins-Daines, David et al. "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices." *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on* 14 May. 2006: I-I.
[67] B. Sekuler, Patrick J. Bennett, Mortimer Mamelak, Allison. "Effects of aging on the useful field of view." Experimental aging research 26.2 (2000): 103-120.
[68] <www.plugincars.com/chevy-volt-photos-video.html> Acccessed 8 June 2015
[69] <http://wot.motortrend.com/tesla-model-x-production-pushed-back-to-late-2014-340275.html/tesla-model-x-dashboard-1/> Accessed 8 June 2015.
[70] <http://soaznewsx.com/Portals/0/Images/Mar%202014%20pics/030314%20pics/Apple-to-launch-in-car-entertainment-system-CarPlay-at-Geneva-Motor-Show.jpg> Accessed 8 June 2015.
[71] <http://www.airlinereporter.com/wp-content/uploads/2012/03/hud.jpg> Accessed 8 June 2015.
[72] <http://commons.wikimedia.org/wiki/File:F-18_HUD_gun_symbology.jpeg> Accessed 8 June 2015.
[73] <http://www.oldcarbrochures.org/NA/Oldsmobile/1988-Oldsmobile/1988-Oldsmobile-Cutlass-Supreme-Pace-Car-Folder/1988-Oldsmobile-Cutlass-Supreme-Pace-Car-02> Accessed 8 June 2015.
[74] <http://i.ytimg.com/vi/AzVV8UUIu48/maxresdefault.jpg> Accessed 8 June 2015.
[75] <http://en.wikipedia.org/wiki/Head-up_display#/media/File:E60hud.JPG> Accessed 8 June 2015.
[76] <http://en.wikipedia.org/wiki/Head-up_display#/media/File:Head_Up_Display_of_a_Toyota_Prius.JPG> Accessed 8 June 2015.
[77] <http://electronics360.globalspec.com/images/assets/160/5160/150323_navigation_image2-fullsize.jpg> Accessed 8 June 2015.
[78] <http://www.sygic.com/gps-navigation/addon/head-up-display> Accessed 8 June 2015.
[79] <https://s3.amazonaws.com/crowdtiltopen/CrowdtiltOpen/uploads/campaigns/facebook_images/000/001/390/original.jpg> Accessed 8 June 2015.
[80] Harkin, Dane, William Cartwright, and Michael Black. "Decomposing the map: using head-up display for vehicle navigation." Proceedings of the 21st International Cartographic Conference 2005.
[81] Ranney, Thomas A. "Driver distraction: A review of the current state-of-knowledge." Apr. 2008.