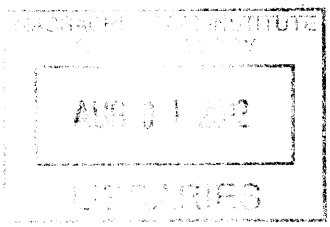
Exploring the human-car bond through an Affective Intelligent Driving Agent (AIDA)



by

Nancy Foen

BSc. Electrical Engineering and Computer Science, MIT (2010)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012

Author_____
Department of Electrical Engineering and Computer Science
January 26, 2012

Certified by_____
Dr. Cynthia Breazeal
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by_____
Dennis M. Freeman
Chairman of the Department Graduate Committee
Department of Electrical Engineering and Computer Science

Exploring the human-car bond through an Affective Intelligent Driving Agent (AIDA)

by

Nancy Foen

Submitted to the Department of Electrical Engineering and Computer Science
on January 26, 2012, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As the amount of time spent inside vehicles increases, there is an ever more pressing need for safe interfaces that support drivers while they are multi-tasking. We present an Affective Intelligent Driving Agent (AIDA), a sociable robot designed to sit at the dashboard of a vehicle and behave as a friendly assistant. This highly expressive robot was designed to receive signals from the vehicle and use an Android phone as its face and main computational unit to manage the information that must be delivered to the driver. AIDA communicates with the driver through speech, imitating the interaction model that exists between a driver and another passenger. This research platform explores the possibility of using a sociable robot as an interface to connect driver and car, creating a stronger bond between them.

Thesis Supervisor: Dr. Cynthia Breazeal
Title: Associate Professor of Media Arts and Sciences, Program in Media Arts and Sciences

Acknowledgments

I would like to thank all those who made this thesis possible.

Foremost, I owe my sincerest gratitude to my supervisor, Cynthia Breazeal, for her inspiring ideas and her exceptional feedback whenever we discussed the design and technical aspects of this work. It has been a pleasure to be part of her group and do research side by side with such talented individuals.

I would like to thank the Volkswagen Group of America for sponsoring me during this project, and the AUDI's Engineering Research Lab for their invaluable guidance through the design stages of this thesis.

I am truly grateful to Mikey Siegel and Trevor Shannon for the work they put into developing the initial prototype of this project, which became the foundation of my work, to Fardad Faridi for his creative character designs and animations, and to Marc Strauss for all his contributions to the new mechanical model.

This thesis would not have been possible without the support and guidance of every member of the Personal Robots Group. Special thanks go to Jesse Gray and Adam Setapen for their help with Android devices, Jin Joo for patiently helping me with the R1D1 software, and to Kenton Williams for being an amazing person to work next to.

Warm thanks go to my friends for their continuous support and for bringing so much joy into my life. Thanks to Maria Rodriguez and Dorothy Brown for keeping my spirits up, and for helping me balance my academic and social life.

I especially want to thank Sigurður Örn Aðalgeirsson for encouraging me during this project, and for kindly offering me extensive technical support.

Finally, and most importantly, I wish to thank my family for always being there for me. Thanks to my siblings, Patricia and Daniel, for being a constant source of laughter, love and inspiration, and to my parents for raising me, loving me and believing in me. To them, I dedicate this thesis, gracias de todo corazón.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem

Given the significant amount of time people spend traveling in their cars, there is a high demand for a pleasant in-vehicle experience that does not compromise the passengers' safety. As reported by the U.S. Department of Transportation, Americans spend about 86 minutes per day in their vehicles, and drivers spend around 55 minutes per day behind the wheel [1]. These extended travel times only make it harder for the driver to disregard other activities and focus solely on operating the vehicle.

Frequently, people tend to multi-task in an attempt to make better use of their time. There are several tasks that drivers want to accomplish while they are traveling, which may include planning a route to their destination, checking the weather and traffic conditions, monitoring the vehicle, making mobile phone calls, exchanging e-mails and SMS (Short Message Service), and dealing with entertainment systems. As a result, it is common to find drivers manipulating their In-Vehicle systems and personal devices while they are operating their automobiles.

Interacting with several applications at once can be an overwhelming and stressful experience. It is possible to find a person driving and trying to look for a suitable song to listen to, while the low gas sign is blinking and the mobile phone is beeping to announce that a new message has been received. Not only must the drivers absorb all of this information at once, but they also have to switch between the different modalities and forms of interactions used by each of the applications they are dealing with. Thus, multitasking usually results in a high cognitive load, which tends to have a negative effect on the driving performance.

According to the Multiple Resource Theory, these effects become worse as the tasks at hand share the same stimulus modality. Hence, a task that requires visual processing resources from the driver would be more likely to have an adverse effect on a person's driving performance than one that requires auditory or haptic responses [16]. Serializing and prioritizing the information is then an attractive concept to keep supporting drivers while protecting their safety. In general, a great deal of care should

be taken when designing in-vehicle user interfaces, to ensure that the actions required from the users cause little interference with the primary driving tasks.

Even if In-Vehicle Technologies (IVT) are well designed to keep the driving experience as safe as possible, it is still inconvenient for drivers to limit themselves to the systems that are part of the automobile. Most users would like to have access to their favorite applications anywhere [9], including the inside of their cars, in spite of the fact that some of these applications were not designed for a driving context. As of today, most vehicles fail to provide a seamless transition for people as they are going into and out of their vehicles [11]. Generally, data does not flow into the car unless additional personal devices are brought in, and used while driving.

1.2 Approach

In an effort to address these challenges and develop a stronger bond between driver and car, the Personal Robots Group in collaboration with Audi has developed AIDA, an Affective Intelligent Driving Agent. AIDA is an expressive, sociable robot designed to sit on the dashboard of a car and behave as a driving companion. The aim of this friendly assistant is to help users accomplish their tasks, without having any control over the vehicle's components that are directly related to the driving task. By using an android phone as its face and main computational unit, AIDA is intended to increase in-car safety by keeping the phone out of the driver's hands. However, given that AIDA is an intermediary between the user, the phone's applications, the vehicle and the environment, people will not lose access to the applications they need to perform their activities. Currently, this robot serves as a research platform where different behaviors can be implemented and tested to evaluate their effectiveness.

1.3 Thesis Overview

This thesis is organized as follows. Chapter 2 presents related work in the fields of prioritizing information, exploring the driver's emotional state, merging mobile devices with in-car systems and the importance of expressive systems. It also introduces AIDA's original design, and motivation, developed prior to the work involved in this thesis. Then, Chapter 3 gives an overview of AIDA's final design and the general features of its new interface, while Chapter 4 explains the design and architecture in detail. Descriptions of the behaviors that were implemented to test the agent's effectiveness follow in Chapter 5, and finally, Chapter 6 provides a summary of the project and a discussion about possible future work.

Chapter 2

Background

2.1 Related Work

Several systems have been developed to tackle the aforementioned concerns. To the best of our knowledge, no interface has been developed to deal with all these issues simultaneously nor has a sociable robot ever been used to explore the emotional bond between driver and car.

2.1.1 Systems that Prioritize Information

There are critical times that require drivers to direct all of their attention to the driving task. Besides, not all of the information that a diver receives when operating a vehicle requires immediate attention. For instance, a driver can usually wait until after he has completed a lane switch to be notified that a text message has been received. Serializing information delivered to the driver, and finding an appropriate time to deliver such notifications, could help reduce the driver's cognitive load, and improve driving performance.

COMUNICAR, which translates to 'communicate', is a European project that developed a Human Machine Interface (HMI) that gathers and manages information, to then deliver it to the driver. This project uses a Communication Multimedia Unit to collect information about the vehicle and its surroundings, including data related to navigation, collision warning, the Internet, entertaining systems and messaging services [2]. Once messages are received, certain algorithms are used to define their level of priority and their Total Level of Risk (TLR). Then a software unit, called Information Manager, uses these results together with context information and possible output modalities to decide what pieces of information should be issued, what is a suitable modality and when is the most appropriate time to deliver them [8].

AIDE (Adaptive Integrated Driver Vehicle Interface) is the continuation of the COMUNICAR project. It also intends to extract information from other driving assistants, perform a priority analysis, and communicate with the user in an organized

way that reduces stress and cognitive overload [3]. AIDE expands on the previous design by taking into consideration the driver's behavior into the decision making process. Also, it was designed to model a true dialogue between the driver and the vehicle [8].

The Human Media Interaction Group at the University of Twente explored a different approach to the problem of driver distraction and cognitive overload. Instead of prioritizing or delaying the notifications, the information was presented upon arrival, but additional cues were included to express the priority level of the message. These cues would allow to driver to decide when to deal with the received message. Their study found that their informative interruption cues (IIC) were learned quickly and identified accurately [10].

Studies and projects focused on the prioritization of the information delivered to the driver make significant contributions by reducing the driver's workload and improving the human-car interaction. This is certainly a common goal that is shared with the AIDA project. However, AIDA also explores the social aspect of the interaction and the possibility of taking proactive actions that reach out to other applications in an attempt to offer more support to the driver.

2.1.2 The Driver-Car Emotional Bond

A driver's emotional state can have a significant effect on a driver's behavior and safety. A depressed, sleepy or stressed driver is not as likely to be fully focused on the road, as compared to a wide-awake driver. On the other hand, an angry or frustrated driver is more prone to road rage, since they are more likely to make risky decisions that could potentially lead to accidents. Therefore, exploring how to affect the driver's emotional state could lead to improvements on driving performance and in-car safety.

A study performed by Clifford Nass and Scott Brave concluded that matching a driver's emotions with the in-car system's tone of voice improves the driving per-

formance. Matching happy drivers with an enthused voice and upset drivers with a subdued voice resulted in a driving performance that was significantly better than the one caused by the opposite combination. Even though happy drivers performed better than upset drivers, the difference was not as significant as the one posed by matching tones [14].

C. Jones explored the possibility of equipping vehicles with a recognition system capable of analyzing speech to identify a person's mood. His Emotive Driver Project demonstrated that an automatic system is able to classify scripted conversations into different emotion categories with reasonable accuracy. There was approximately an correlation between the recognition system's classification and the categorization made by a human listener [15].

An interface that could identify the driver's mood and react accordingly could potentially improve in-car safety and the human-car interaction. It is the belief of the author, that designing systems capable of delivering information in an expressive, sociable way could also improve the quality of communication between the driver and the car. If the vehicle is able to express itself in a way that feels natural and familiar to the driver, then there could be a deeper understanding of the messages conveyed by the car.

2.1.3 Mobile Devices in Vehicles

In general, care must be taken when designing in-vehicle systems to guarantee high quality user experiences that keep the level of distraction low. However, avoiding distraction is not an objective when developing applications for mobile devices. On the other hand, they are usually designed to provide as much entertainment as possible, requiring high levels of attention. Yet, users generally bring their mobile devices with them into the vehicle, and use their applications even while driving. Because the manipulation of personal devices distracts drivers from the main driving task, it tends to have a detrimental effect on their performance.

Consequently, much work has been done for drivers to access their mobile phone applications without compromising safety. Car docks allow users to mount their mobile devices in the vehicle's dashboard or windshield. These systems are convenient, but they still degrade user experience and safety because it is difficult to manipulate a phone' small screen and the applications still have a very distractive nature.

A different approach has been to integrate mobile applications into the car In-Vehicle Infotainment (IVI) Systems. This option allows users to access some of their phone applications through the IVI systems' interfaces. There are systems that only integrate phone conversations, while others, like Terminal Mode [9], handle a wider range of applications. In the last case, mobile phones run all services and applications, while the IVI system is responsible for all input/output functions.

Alternatively, the solution can be incorporated directly into the phone. On October 4th, 2011, Apple Inc. released its Siri feature on the iPhone 4S [13]. Though it was not developed specifically for the in-car context, this feature can manage applications, thus reducing the driver's cognitive load. Meant to be a personal assistant, Siri is designed to receive the user's voice commands, spoken naturally, to access and manage other applications. Among its capabilities are: setting remainders, finding directions, placing calls, sending messages, and scheduling events [13]. One of its most salient features is its mode of interaction, which models everyday conversations and allows scenarios like the following:

User: "Is it going to be chilly in San Francisco this weekend?"
Siri: "Not too cold... maybe down to 61 in San Francisco"
User: "What about Napa Valley?"
Siri: "Doesn't seem like it; it won't get below 68 in Napa."

In general, a considerable amount of effort has been put into finding ways to make the manipulation of mobile devices safe while inside a vehicle. However, most of the resulting systems remain highly reactive, and the driver still has to initiate all of the tasks needed to obtain the desired information. In our opinion, not enough research has been done on how to create interfaces that take the initiative to take actions and

perform searches that could potentially help the user. These systems could make the information that the driver needs readily available, providing additional support and reducing stress and cognitive load.

2.2 Original Design

2.2.1 Initial Motivation

Spending a significant amount of time behind the wheel on a daily basis can become a stressful and monotonous experience, especially for drivers who are alone in their cars. What is more, driving is not a trivial task, and the passengers' safety can be compromised whenever the driver has a high cognitive load due to the multiple signals emitted by the vehicle. The idea of a personal, robotic, sociable assistant was originated in an effort to provide some emotional support, while reducing the amount of attention that is required by warnings inside and outside the car.

One of the main objectives of this project was to equip the vehicle with the ability to express emotions and communicate in sociable ways. Mikey Siegel, Personal Robot Group alumni, together with animator Fardad Faridi, devised and developed AIDA's first prototype and the ideas leading to it. Initially, much thought was put into integrating the expressiveness directly into the car's dashboard. Some of the early concepts attempted to make the dashboard more dynamic by using flexible skin to make eyes out of the speedometer and tachometer, or by building the entire dashboard out of a malleable membrane that allows it to change shape depending on the situation. Both ideas are depicted in Figure 2-1 and Figure 2-2 respectively [12].

Other concepts explored the possibility of having expressions in a more abstract form, by putting a high-resolution, multi-touch display within the vehicle's logo. Having different lighting and animation patterns would enable the car to provide a wide range of messages, and it will also allow the expression of feelings. However, because this is not a human-like mode of communication, it may be unfamiliar to the drivers

Figure 2-1: Expressive Eyes Merged Into The Vehicle's Speedometer and Tachometer [12]

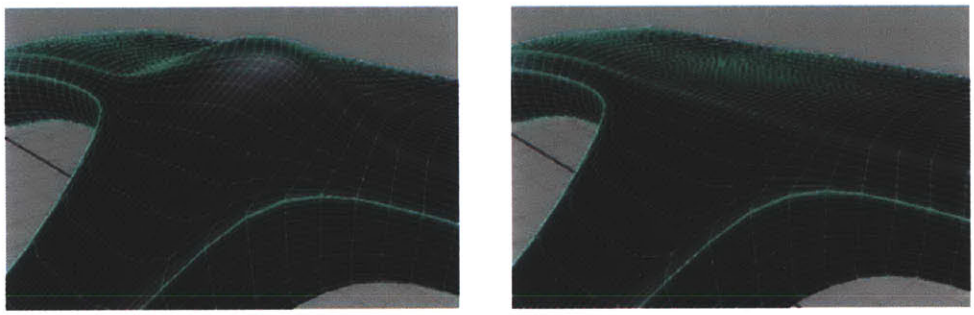


Figure 2-2: Dynamic Dashboard Using a Flexible Membrane [12]

and it may take some time to get used to. Therefore, a more anthropomorphic design was chosen, a small robot whose head and neck could came out of the dashboard.

2.2.2 First Prototype

Facial expressions and body language play a crucial role in the interactions between humans, and even between humans and animals. Generally, we can tell whether others are happy, sad or angry simply by looking at their faces and the way they move around. Because of this, it was considered that AIDA would be better suited to express itself if it was equipped with a head, a face, and a couple degrees of freedom. Accordingly, AIDA's first prototype consisted on a mechanical head and neck, equipped with five degrees of freedom that enabled the robot to look around, nod, rise from the dashboard, and use body language to express feelings like being

curious or happy.

To complement the neck movements, facial expressions were incorporated in the design. The robot's head was a plastic 3D-printed shell with a laser projector embedded inside, used to display face animations. A complex two-mirror reflection system was devised to project the expressions successfully. The completed prototype is illustrated in Figure 2-3. To test the developed hardware, two behaviors were implemented in software: face tracking and playback of preset animations. Because the software that determined the movements of the motors and the face expressions ran on an Apple Mac Mini, the robot had access to any input and output that were available to this computer, including microphone and speakers. However, none of these devices were used on this version of AIDA.

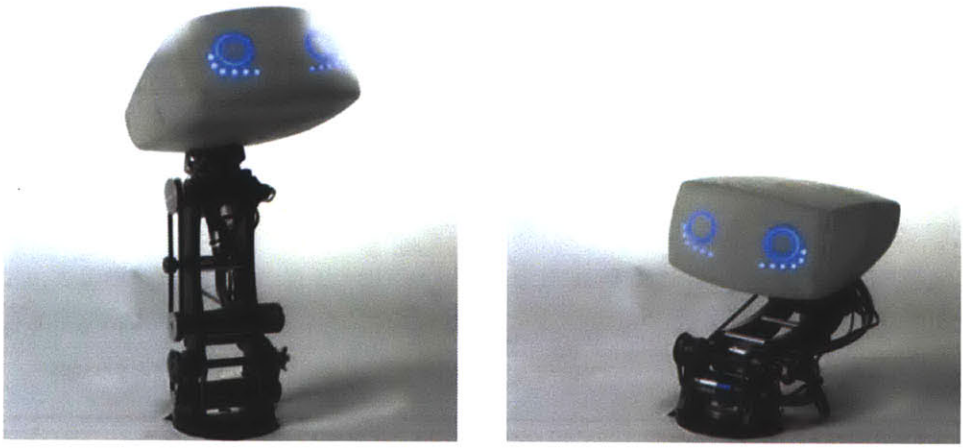


Figure 2-3: AIDA's First Prototype

Chapter 3

Final System Overview

3.1 Expanding on the Original Design

The original prototype was functional and had the potential to support sophisticated, sociable behaviors. However, before we got involved into the development of the software behind it, we decided to revise and improve the previous model. The main concern with the initial design was the intricacy of the projection system embedded in the head shell. Although the display was aesthetically pleasing, the projector was heavy and expensive, and the overall system required considerable effort to replicate. Moreover, some of the movements seemed jerky, probably because the weight on the head was meeting or surpassing the limitations imposed by the motor's stall torque.

Consequently, we explored other alternatives to replace the robot's head and face. Two important objectives were to provide additional support to the driver, and to keep the same face animations that characterized the original design. In order to achieve the first goal, it was imperative for the new design to make the driving experience even safer and more comfortable. Although it is true that multiple signals from within the car result in a higher cognitive load, distractions can also be due to drivers using their mobile devices while operating their vehicles. Thus, it is to be expected that expanding AIDA's capabilities to manage phone applications would reduce drivers' stress and it would contribute to their safety.

Given the large screens, high-resolution displays and multiple video formats supported by various phones nowadays, playing the animations that were previously used, and even more complex animations, would not be unfeasible. Reusing the facial expressions would allow us to preserve AIDA's persona, which was the second objective of this new version. Therefore, it was decided that a phone should be used as the robot's face. With this, AIDA could transition from being an interface just between the driver and the vehicle, to being an interface between the user, the vehicle, mobile applications, and their surroundings, both inside and outside the car.

Essentially, the second design was developed using a phone as its core-processing unit, in an effort to provide an effective and proactive assistant, capable of replicating

social behaviors and improving in-car safety. Currently, AIDA is a flexible research platform in which new and diverse concepts can be easily implemented and tested. However, some initial behaviors were developed and included in the current design to incorporate some features that we considered to be important.

3.2 Character Overview

The behavior and support offered by another passenger in the front seat was used as a role model to defined the desired characteristics of this new agent. We decided that AIDA should be a centralized source of information that assisted the driver, everywhere and at anytime, through expressive and sociable actions. This section describes these qualities in more detail.

3.2.1 Centralized Source of Information

Currently, users must switch between different applications and devices to obtain the information that they need. Our goal is for AIDA to be a proactive agent, capable of having a two-way communication with the driver and passengers. AIDA can receive commands and feedback from the users, and return relevant information that may be required by them to accomplish secondary, non-driving tasks. This information could be extracted from the vehicle's sensors, the Internet and the phone's applications. AIDA could also deliver information that has previously been added to its database by its users, or its designer.

To avoid high cognitive loads and unnecessary levels of stress, the information delivered should be prioritized and serialized. For instance, instead of simultaneously displaying a zip-belt warning, a remainder of the next event in the driver's calendar and an incoming text message, all of these signals could be directed to AIDA to be delivered at an appropriate time, in a natural way that feels familiar to the users. Upon the arrival of all notifications, the information can be ordered with respect

to its priority level or by preset rules, and it can be serially delivered later. Visual messages, like a map or an animation of a warning sign, can be used to provide a clearer, reinforced message.

3.2.2 Assistant Role

All of the implemented behaviors should contribute to its sociable and supportive nature. AIDA is meant to be a friendly driving companion that facilitates the completion of tasks for those riding the vehicle. It is common for a passenger to be helping the driver find a route to their destination, reading and sending e-mail and SMS messages, looking for suitable entertainment, and informing the driver about points of interest around the area. This is the interaction model that AIDA is trying to reproduce by becoming another car passenger that supports the driver, especially when he is alone.

However, passengers need to know a lot of information about the driver, the environment and the vehicle if they want to become effective assistants. Even though a passenger can successfully send a text message after the driver has recited the recipient's phone number, it would be much more efficient if the passenger already had access to the driver's contact list. In this same way, AIDA should not be limited to receiving signals from different sources; it should have access to the information it needs to help the driver in an effective and practical manner. This goal could be accomplished by utilizing the phone that will be integrated to the overall system.

Having a responsive agent who could fulfill requests given by the driver will probably upgrade the quality of the driving experience. Nevertheless, we envision AIDA to have a more proactive role, taking the initiative to make decisions and take actions that could assist the driver even if they have not made an explicit request. In a situation where a driver receives a message from friends to meet at a certain time and place, AIDA could potentially check the user's calendar for availability and offer to send a generic reply saying whether the driver can or cannot make it.

3.2.3 Expressive Nature

The relationship between driver and car can be further expanded by adding an additional channel of communication between them, one focused on social interactions. Humans are inherently sociable beings, constantly expressing and perceiving emotions. Whenever people try to express themselves, they usually employ a wide range of body movements, facial expressions, and tones of voice. Even when dealing with animals, a feeling of deeper understanding could result from the recognition of different moods from certain movements, expressions and sounds. It is only natural to expect similar social behaviors from interactions with others beings or devices, especially if they have some anthropomorphic features that resemble other living creatures.

The AIDA project is trying to expand this concept, incorporating it to the relationship between driver and car. Through AIDA's use of facial expressions and physical movements, we expect drivers and passengers to have a better understanding of the messages that the car is trying to convey, developing a stronger human-car bond and a deeper sense of identification.

3.2.4 Availability Everywhere

Even though the main focus of this project is to improve safety and the quality of the user experience while inside the car, the phone application should also be available outside the vehicle. This increases the amount of interaction time between AIDA and the driver, hopefully fostering a closer relationship between them. As users invest more time on AIDA, changing its settings and adding favorites, the level of personalization of the agent increases. The greetings and suggestions would then be different for each individual application, as they would depend on the user's preferences. Even though this amount of personalization may be enough to stimulate a stronger bond, more personalization could be achieved with a feedback system, in which the robot could learn from the user's reactions. This will enable AIDA to know which behaviors cause a positive response and which behaviors do not. As a result,

the accepted behaviors would be repeated more frequently, and the rejected behaviors would be chosen with less frequency.

Besides having the opportunity to personalize AIDA, there are other benefits to an extended interaction between the agent and its owner. One of these benefits is that AIDA could be useful to the driver even if a vehicle is not being operated. Since AIDA's main computational unit is the phone, the application could be accessed anywhere, and it can therefore provide information and notifications even if the user is not in a car; It could take automatic actions like providing event reminders to a user who is still at the office.

Given that AIDA and the applications it uses are available everywhere, users would have a more seamless transitions as they go into and out of their vehicles. AIDA would facilitate this transition by being a constant agent that accompanies the user regardless of their location. Also, this assistant would avoid redundant actions from the user side, like changing a meeting address both in the calendar and in the destination field in a navigation system. The applications that AIDA manipulates, like the user's contact list and calendar, are being accessed directly and any modification made to them would automatically be available to this agent. Because of this, data flows into and out of the vehicle without the user having to do any additional searches or updates.

Chapter 4

System Design

4.1 Introduction

As aforementioned, AIDA's main computational unit is a phone. This device carries an application that is responsible for all the communication between the driver, the car, the physical agent, other phone applications, and outside sources of information. Currently, the system consists of a physical robot, an Android phone with a customized application and an external computer. These components interact with each other to obtain any necessary information, decide what behavior to do next, and then carry out all the actions chosen.

4.2 Robot's Hardware

AIDA's physical system is composed of a robotic head and neck that could be mounted on top of a vehicle's dashboard. The motors that control the mechanical body are put in motion by commands given by an external computer which will be connected to the robot's circuit boards via USB. However, the main channel of interaction between the system and its user is a phone attached in the front side of the head to display the face of the robot.

4.2.1 The Phone and Face of the System

There are certain benefits for using a mobile phone as the robot's face. One of the main advantages is the portability of these devices, which allows users to keep their phones on them all the time. Moreover, mobile phones already contain a lot of information about their users though applications like the directory, the calendar, and the music player. For this project, we decided to use an Android phone to take advantage of its popularity, low development cost, open-source platform and its rich hardware and Java support.

Popularity and Ease of Use While Androids are the world's best-selling smartphones [4], they also have low development costs due to fact that there are no licensing fees or expensive development tools. In fact, given the extensive set of Java libraries supported by Android and its comprehensive Software Development Kit (SDK), it is fairly easy for Java developers to create or expand an application, without any prior Android experience. In AIDA's case, this is a valuable advantage since it is possible for others software developers to expand its application by designing and implementing new behaviors.

Java Support The fact that the Android operating system provides Java Support was also beneficial for us to develop the software behind the basic AIDA design; it was relatively straightforward to port some of the existing code into an Android platform. The majority of the robots developed at the Personal Robots Group make use of a code base written in Java, currently called R1D1, which is described in Section 4.3.2.

Phone's Additional Hardware Moreover, any application developed in Android has access to the additional hardware and peripherals available to the phone. Some of these include cameras, Global Position Systems (GPS), microphone, touchscreens, accelerometers and proximity sensors.

Access to More Information about the user The android in AIDA's face already has access to all sorts of data that drivers have stored in it because of their daily interactions with their phones, e.g. contacts' information, calendar events and music. Users also have the option to add additional information about what they like on the AIDA application itself. At this time, they can include their favorite restaurants, artists, and gas stations. AIDA could then make suggestions that are more in tune with the driver's preferences. For instance, if it finds several gas stations in the area, it would give preference to the brands that have been inputted in the 'Favorites' list.

Media Support One of the objectives was to preserve the same face design and expressions that were used on the original design. The original videos projected on the face were in a QuickTime File Format, which can be conveniently converted into both MP4 and 3GP formats. Fortunately, Android phones support these file formats.

All these factors contributed to the creation of a powerful, yet flexible, AIDA Application that became the main interface that connects the driver with other services and sources of information, including other phone applications, car signals and web pages.

4.2.2 Android Epic Galaxy X

Once the decision to use an Android phone was made, the next step was to pick the phone model that would be used to test the behaviors that were going to be implemented. The software development stage was scheduled to start on January 2011, so it was essential for the phone to have been released by the end of the year 2010. Given the variety of Android phones available at that time, we used the following criteria to filter the possibilities:

Screen Size and Resolution In order to display vibrant animations and to transform the phone into a believable face for a friendly character, the screen required a minimum size and resolution. The limit chosen were 90mm for screen size and a resolution of 480 x 800 px.

Assisted GPS Providing effective assistance under driving conditions these days frequently depends on having the ability to locate the user's position quickly and effectively. Sometimes, finding the current location and the route to follow a few seconds too late can cost the driver valuable time by forcing them to take alternative paths. We chose to have Assisted GPS, a system that uses network resources in addition to satellite signals to improve the startup performance.

Front Facing Camera (FFC) At the beginning of the design process, we envisioned AIDA to have some face tracking and face recognition capabilities. Installing a mood recognition system was also desirable, as it would have enabled more personalization and emotional support. Each one of these functions required a FFC considering that the screen had to face the user. Because of the limited number of models that were equipped with a FFC at the time, this requirement became an essential and helpful filter.

Accelerometer This additional hardware would be required to correlate AIDA's behaviors with the circumstances surrounding the driving environment. For instance, if the driver was speeding up, or making complicated maneuvers, then AIDA could postpone the delivery of any messages for the driver to focus on operating the vehicle. Therefore, we only considered phones that had accelerometers in it.

Keyboard Certain phone models have a physical keyboard fixed on the phone. In these cases, the keyboard takes a significant amount of space, thus reducing the screen size available for display. The phone orientation was intended to be horizontal, and in our opinion, the AIDA character would be more believable and it would have a closer resemblance to a creature if it did not have a set of keys on the side of its face. Due to these aesthetic reasons, we constrained our search to phones with touchscreens, or hidden keyboards.

Weight Having a load limit was a reasonable constraint because the head would be controlled and lifted by several motors, each one with a stall torque. The limit was set at 200g. More details on torque related calculations are shown in Section 4.2.3.

After applying these filters, we obtained the following possible phone: HTC Evo 4G, Samsung I9000 Galaxy S, Samsung Epic 4G and T-Mobile MyTouch 4G. Pictures of each one of these models, in the aforementioned order, are shown in Figure 4-1.

Figure 4-1: Android Models considered to serve as AIDA's face. From left to right: HTC Evo 4G, Samsung I9000 Galaxy S, Samsung Epic 4G and T-Mobile 4G [5]

The phone selected was the Epic 4G. One of its advantages is the convenient balance between its screen size and its weight; the 101.6mm screen has a relatively large size given its weight of 155g. Besides, this phone has a Loudspeaker, could support up to 32GB with a microSD memory card and it lasts up to 500 hours in stand by mode and 5 h 30min while talking. Once the decision was made and the specifications were known, the next step to take was to make the pertinent modifications to the previous prototype, in order to guarantee that the robot operation was appropriate.

4.2.3 Head and Neck Mechanism

The robot physical design, preserved mostly from the first prototype, was carefully designed to resemble an expressive assistant; it was given certain anthropomorphic features, including a head, a neck, and a face. Whenever the robot is not in use, it is meant to sit seamlessly in the dashboard. However AIDA has the capability of fully extending its neck to convey a stronger sense of urgency and excitement, as shown in Figure 4-2. All intermediate positions are available to communicate messages with more meaningful expressions.

Figure 4-2: AIDA's place in the dashboard

The system is equipped with five degrees of freedom:

- Neck Up/Down
- Neck Left/Right
- Head Up/Down
- Head Left/Right
- Head Tilt

These degrees of freedom make the robot's movement highly expressive: Rotation about its neck and head would be useful when conversing with other passengers; the up and down movement of the neck and head can be adjusted according to the priority or intensity of the message; the head tilt would reinforce certain emotions, like sadness or confusion. We anticipate that a sociable, emotive robot would be easier to understand and it would strengthen the bond between the driver and the automobile.

Head Modification

The first prototype had a white plastic head shell with an embedded projector to display facial expressions as shown in Figure 2-3. However, the new version discarded the projector system. Instead, it required a mechanism to safely incorporate the Epic Galaxy X Android phone that would serve as the robot's face. Initially, we were contemplating the possibility of creating a universal design that would be able to hold almost any Android phone. There are several devices that accomplish this task already and are commercially available, like the ones shown in Figure 4-3. The challenge would then be to seamlessly incorporate the commercial device into the head shell built for AIDA.

Even though integrating a universal holder was a feasible option, there were concerns with the multiple looks that the robot could potentially have, given the different dimensions of the phones that could be attached to its head. The simplest head shell design would be a rigid shell that could not be molded according to the phone used, which would not be ideal if the phone length did not match the head width. Assuming a horizontal orientation, two problems could arise, as depicted in Figure 4-4: either the sides of the phone would be suspended, beyond the edges of the face, or the phone would not completely fill the front side of the head.

Figure 4-3: Universal Stand Holders. From left to right: iCarpus01, iCarpus02 and RAM X-Grip [6]

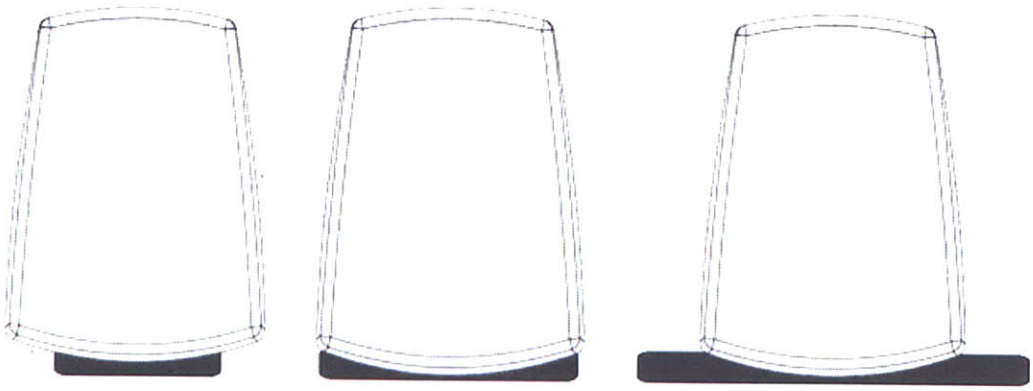


Figure 4-4: AIDA's head as seen from above with phones of different dimensions. Each phone is represented with a gray rectangle. The ideal look is shown in the middle.

It was decided that we would avoid the issue of mismatched dimensions by creating a head shell that would only match a single phone model, specifically, the Epic Galaxy X that was chosen for development. At this time, AIDA is only meant to serve as a research platform to test certain behaviors, so there is no need to have a more flexible holding mechanism. If this platform were to be commercialized, then a head with a universal holding mechanism should be designed to be able to accommodate different phone models.

The general idea was to keep the neck's mechanical and electrical design intact, changing the model simply by switching the head shell to hold the phone. Given that the phone is relatively heavy and it is located precisely at the edge, on the front side of the head, it exerts a considerable torque on the motors. Therefore, the motors' stall torque placed length and weight constraints on the new model. After performing the calculations shown in the Appendix A, we determined that the motors could hold a recommended weight of 200g, and the recommended head length was 7.5cm.

Accordingly, the new head model had a length of 12cm, with a distance of 7cm between the phone and the motor joint. In addition to changing the dimensions, we also substituted the white plastic material used on the previous prototype. Since the projection system was being replaced, the translucent shell was not necessary

anymore. Instead, we selected a glossy black ABS (Acrylonitrile butadiene styrene) material, which we thought would not stand out as much. In our opinion, matching the robot's color and look to that of the dashboard would contribute to the notion that AIDA is a component of the vehicle, and not a completely separate entity. The final model, by Marc Strauss, is illustrated in Figure 4-5, together with the fabricated prototype.

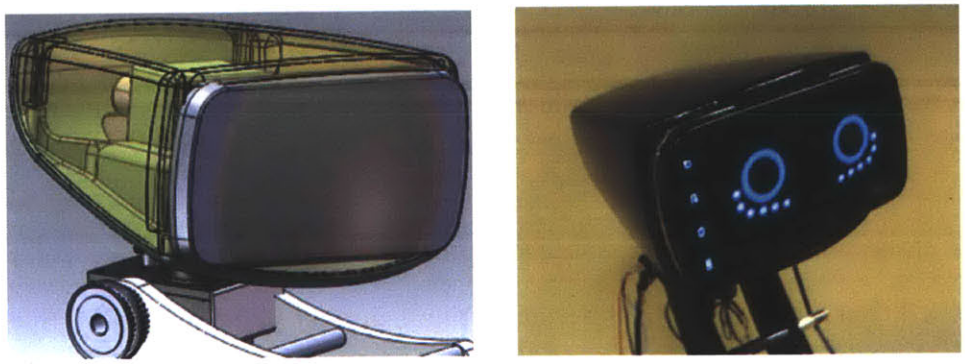


Figure 4-5: AIDA head model and finished prototype. Designed by Marc Strauss.

4.3 Software

Adjusting AIDA's hardware to incorporate an Android phone was mainly a matter of redesigning the head shell and confirming that the electrical components were capable of controlling the new model. However, given that the new design aimed to use an Android phone as the main computational unit, the software used by the previous prototype had to be altered substantially because it was centered on an Apple Mac Mini computer. Above all, the architecture had to be modified to transfer all input and output functions to the phone, making it the interface between the user the rest of the elements of the system.

Besides redefining the way the components were organized, we also had to expand the capabilities that AIDA had by creating and implementing new behaviors. The first prototype was only programmed to track and follow faces, using a camera that

was attached to the base of the robot, below the neck. In this new version of the system, we visualized AIDA behaving as an effective assistant, hence its actions had to be more closely related to the driving context. This agent would provide valuable information to the driver, at an appropriate time, by making use of the available resources, e.g. the user's calendar and contact list, websites and Google Maps.

Essentially, the new version would have an Android application, responsible for communicating with the driver, accessing other applications and deciding the robot's next actions and behavior. Then, there will be another layer, called R1D1, receiving the application's requests and sending the appropriate commands to a controller, in charge of manipulating the mechanical body of the robot. This section describes the new architecture in detail; Figure 4-6 shows a diagram of the overall system with the main components, and the way they are connected to each other.

4.3.1 Controller

At the lowest level of this software implementation, there is a python script that runs in the external computer and behaves like a controller, receiving requests to move the motors to particular positions. The requests received contain the desired motor values at a given time, which were calculated by the R1D1 layer in the system. Once these values are obtained, the controller uses serial communication with the micro controller circuit board, via USB, to set the positions of the five motors. The feedback returned by the incremental motor encoders allows for relatively precise manipulation.

Besides having the crucial role of serving as the intermediary between the application layer and the physical robot, this script also makes sure that the motor movements stay within reasonable limits. Even though the motors are mechanically capable of inducing 360-degree turns, fully rotating most degrees of freedom, a full rotation would look unnatural for any of the joints and should be avoided. For instance, when humans are facing forward, they can turn their head about 90-degrees to the right or to the left, but it is impossible to do a full rotation to return to the

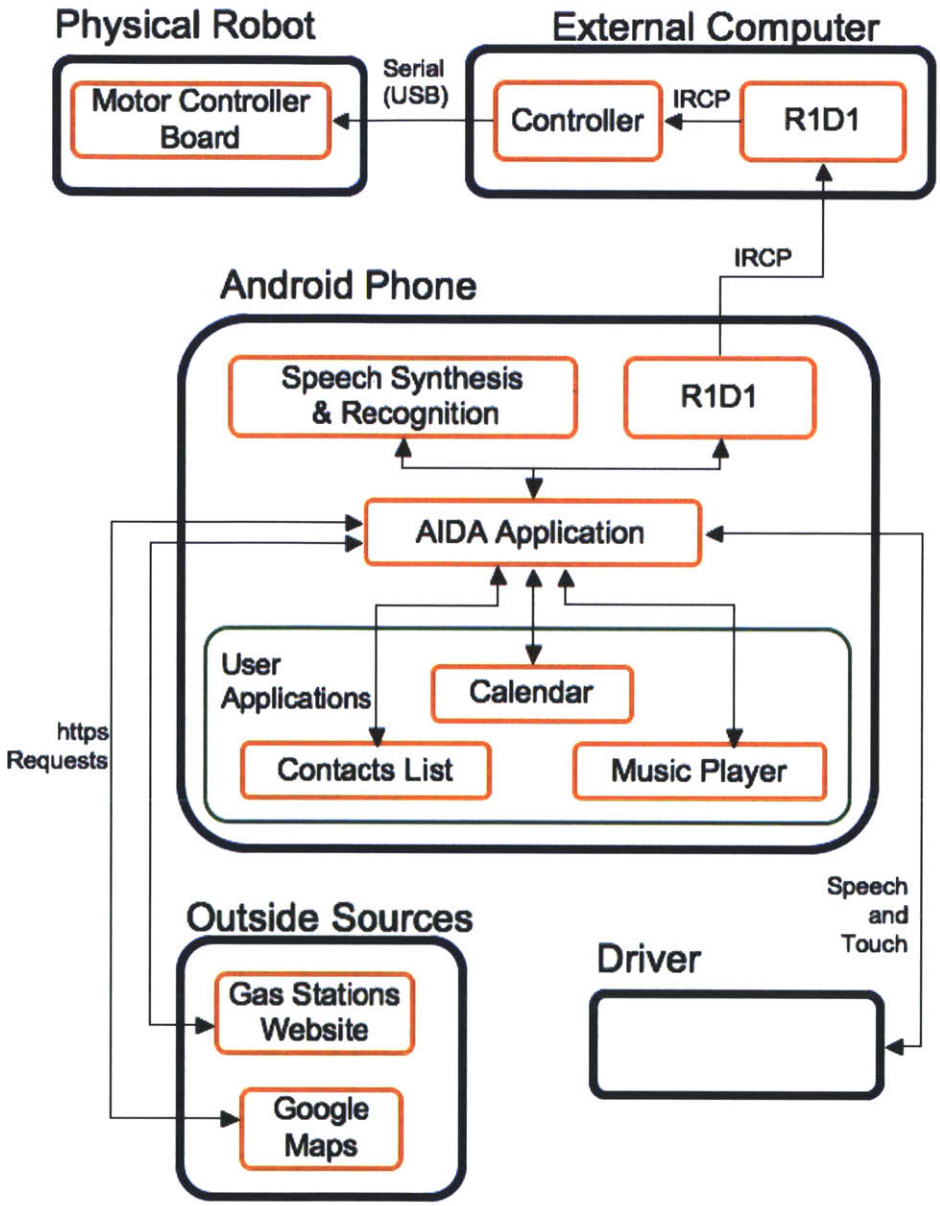


Figure 4-6: AIDA's Architectural Design

original orientation. In view of that, AIDA's movements should be restrained by similar limits.

Likewise, there are also velocity constraints that have to be imposed to guarantee the safety of the robot, the credibility of its movements and the comfort of the user. Very sudden motions can startle the drivers and pose a danger for people inside and

outside the vehicle. Therefore, the python script has a description of each motor, containing more than just the necessary parameters. At this time, it has information about gains, the channel and mode of communication, and it also includes sensible restrictions, like the maximum, minimum and center positions, and the maximum velocity.

4.3.2 R1D1 Code Base

AIDA's software implementation makes use of R1D1, a Java code base developed at the Personal Robots Group and constitutes a substantial architecture used to design synthetic brains for robots. Originally, this code base was only used to control virtual characters, but its capabilities were later extended to handle embodied systems. Even though this architecture is capable of achieving sophisticated behaviors, only a minimal part of it is being used in this project, merely to transfer information between the different components and to create the commands that will be used to control the motors in the mechanical system. There are two versions of R1D1 that must be running to make AIDA work, one in the Android phone and one on an external computer. Both devices must be connected to the same network to enable communication between these platforms.

R1D1 on The External Computer

The R1D1 program running on the external computer is responsible for devising and sending the commands that will be used by the Controller to manipulate the physical robot. Although this code base could collect data, analyze it, decide and carry out actions based on intelligent, social behaviors, it is currently being limited to perform only the last step. In essence, this component receives a command, like 'do transition to happy', and decides, through a Motor System, what motor movements are required to fulfill the request. After all motions have been decided, this program sends the commands to the network, for the python controller to receive it and then transmit

it to the motor controller board located in the robot, via USB. The motor controller would then activate and regulate all physical movements. More details about each one of the java classes involved are given in the following subsections.

AIDA Class The AIDA class defines the creature that will be placed in the virtual world. This class does not include any details about the physical appearance of the robot's mechanical system, but it is linked to the Motor System described below. Used in combination with a 3D model, this class is generally used when developing new behaviors and animations, to test all movements before they are programmed into the real robot.

AIDAMotorSystemMerged Class The Motor System is a class that is constantly being updated and takes care of two crucial tasks: determining what the desired actions and animations are at a particular time, and playing videos of the appropriate facial expressions.

Deciding what are the pertinent motor actions is achieved by mapping a set of given states to predefined joint animations and positions. For instance, a sad state is currently matched to an action that lowers the head and tilts it slightly. However, as the amount of degrees of freedom increases, it becomes more complicated to define the positions of all joints with a single state. In a more general case, a robot's bored state could define the motions of the head and neck, but it does not necessarily need to have information about whether the robot is standing, moving or sitting. To avoid having three separate states ("bored-walking", "bored-sitting" and "bored-standing") we could have two systems, one that defines the mood ("bored") and another that defines the body activity ("walking", "sitting" or "standing"). Then, numerous combinations would be possible with only a few states.

Following these notions, AIDA has two main systems that are affecting the body movements simultaneously, an Idle System and an Emotion System. The Idle system is simply playing a long resting animation in a loop, mostly involving very smooth

movements that simulate involuntary, natural actions like breathing and looking around. This behavior usually dominates when the robot is not showing any particular emotion, and is meant to keep AIDA from having an inanimate appearance.

On the other hand, the Emotion System has several states that can be triggered according to external circumstances. As indicated by the name, these states are closely related to different moods or emotions. Currently, AIDA has been equipped with animations that can represent a robot that is sad, bored, disoriented, happy, surprised and in a neutral state. To avoid abrupt interruptions and switches between these animations, each state is represented as a node, and transitions are limited to the links that have been established between these nodes.

The map that defines AIDA's possible nodes and transitions is relatively simple and is depicted in Figure 4-7. As shown in the figure, because there is no direct link between "Sad" and "Happy", the robot needs to go back to the "Default" node first if it wants to transition from one emotion to the other. This allows for more fluid movements and it facilitates the expansion of the map, given that adding an extra node would only involve two transitions, to and from the "Default" node, instead of requiring a link to each one of the existing nodes.

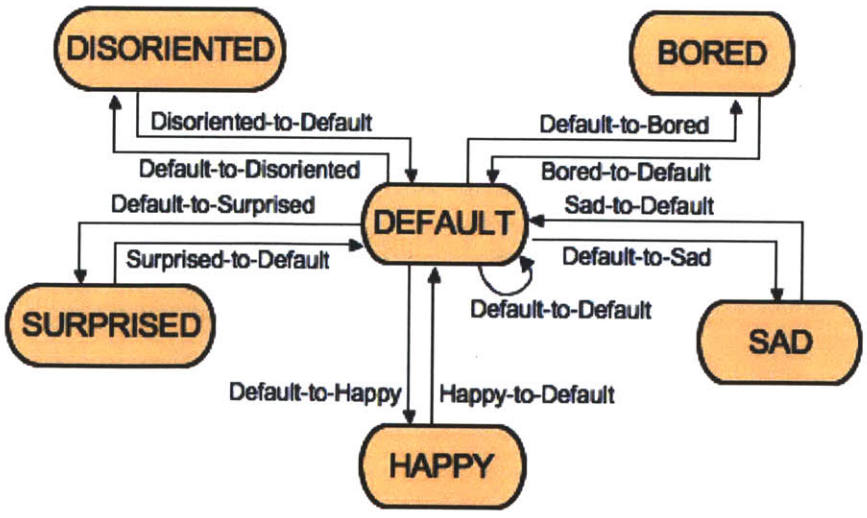


Figure 4-7: Motor System Diagram

The nodes used by the Idle and Emotion Motor Systems are not meant to describe very specific situations. Instead, they are supposed to describe a general mood, so most movements are fairly subtle and applicable to a broad range of situations. However, there are some cases that need very specific actions, like the initial greeting when the driver gets into the car. These animations tend to be sharper and are played by an inner class, called RealtimePlayback, which returns the robot to the default state and then overtakes control of all of the degrees of freedom.

The second task of the Motor System is to play the videos that will be displayed by the Android phone in order to verify that the body movements are compatible with the facial expressions. Except for the blinking videos, usually played every 10 seconds, the rest of the videos are chosen in accordance with the current emotion being acted by the Emotion Motor System; an inner class, named FaceManager, inquires the state of the Motor System, loads the corresponding video file, and plays it frame by frame. Further details on the video expressions are given in Section 4.3.4.

AIDAController Class The AIDAController is the most fundamental class in the R1D1 program that runs on the external computer. This controller, which is different from the python controller, puts all the R1D1 components together and must be launched every time the robot is being used. In general, this controller listens to the requests that are broadcasted by the Android phone and manages its resources to move both the virtual and the physical robot as directed by the phone application.

Once the controller is launched, it begins by setting up the virtual environment, creating a window and placing the robot model in the middle of it. Some of the steps that must be taken to achieve this are: loading the robot's geometry and shading, setting the camera parameters, creating an instance of the AIDA class which is linked to a Motor System, and assigning a Degrees of Freedom Manager. After the model is loaded, its default motor values are set to resemble a low, sitting-like position. Sliders are created to change the neck and head default positions once the application is

running. The virtual 3D model is portrayed in Figure 4-8 in several positions; starting with the default position, all pictures are part of the waking up animation, except for the one on the bottom right.

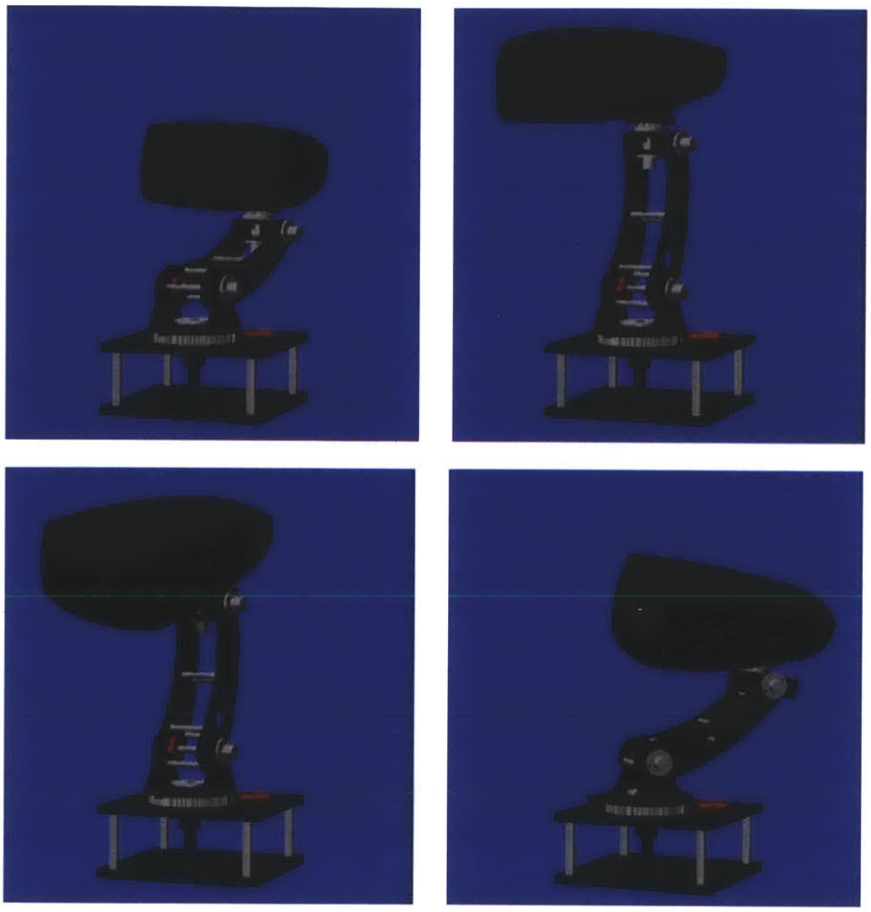


Figure 4-8: AIDA 3D Virtual Model in several positions

After the model is created, the next step is to establish an IRCP (Internet Relay Chat Protocol) Packet Manager, which will be constantly listening to the messages that are broadcasted specifically by the AIDA phone. Packet Identification numbers and IP (Internet Protocol) Addresses are used for this purpose. After a message is received, it is processed by a switch-statement that calls different methods depending on the content of the packet. Currently, each packet received contains an integer, meant to be a unique ID that represents the request for a specific action.

This switch-statement structure was devised keeping in mind that AIDA is meant to be a platform used to test new behaviors, so it is necessary for it to be extensible. In the case of packet handling, incorporating a new command simply requires the definition of a new action ID and the creation of the movements that must be matched to such command. Also, because there is a predefined order in which each case is tested, it is possible to establish a priority system, attending to certain messages before others. For instance, a surprised mood request is handled before a happy mood request.

Integer IDs can currently place a request for either a specialized animation or a general mood transition. Whenever the phone application requests a specialized animation, the controller calls a Motor System method directly, starting the appropriate RealtimePlayback instance described in the previous section. The controller window contains buttons that trigger these animations for debugging purposes. Once the selected animation is over, the system returns control of the joints to the Idle and the Emotion Motor Systems.

On the other hand, an inner class called the EmotionManager handles the general mood transitions, also employing a priority system to travel between emotion nodes. This manager can stack several requests, complying with them in a serial manner, following the paths illustrated in Figure 4-7. Each time a mood is requested, the Emotion Motor System takes full control of the degrees of freedom needed and switches to the desired mood. In order to visualize and debug the animations involved with these transitions, we have added several buttons on the controller that can be chosen one at a time to manually override the current emotion and simulate the change of moods.

Another useful window that is part of the controller is the enabler of the main motors. Every time a new animation is created, it should be tested before it is implemented in the physical robot in order to protect the motors in the robot. This testing stage is necessary to verify that the animations files actually result in the

intended movements. Therefore, by default, the AIDA java controller does not try to communicate with the motor controller board even if it is connected via USB. If the robot is powered, connected to the computer, and the python script is running, then the safety monitor can be overridden and the joints that are needed can be enabled. A picture of what the controller looks like, with some of its components labeled is portrayed in Figure 4-9, and the process of enabling joint is shown in Figure 4-10.

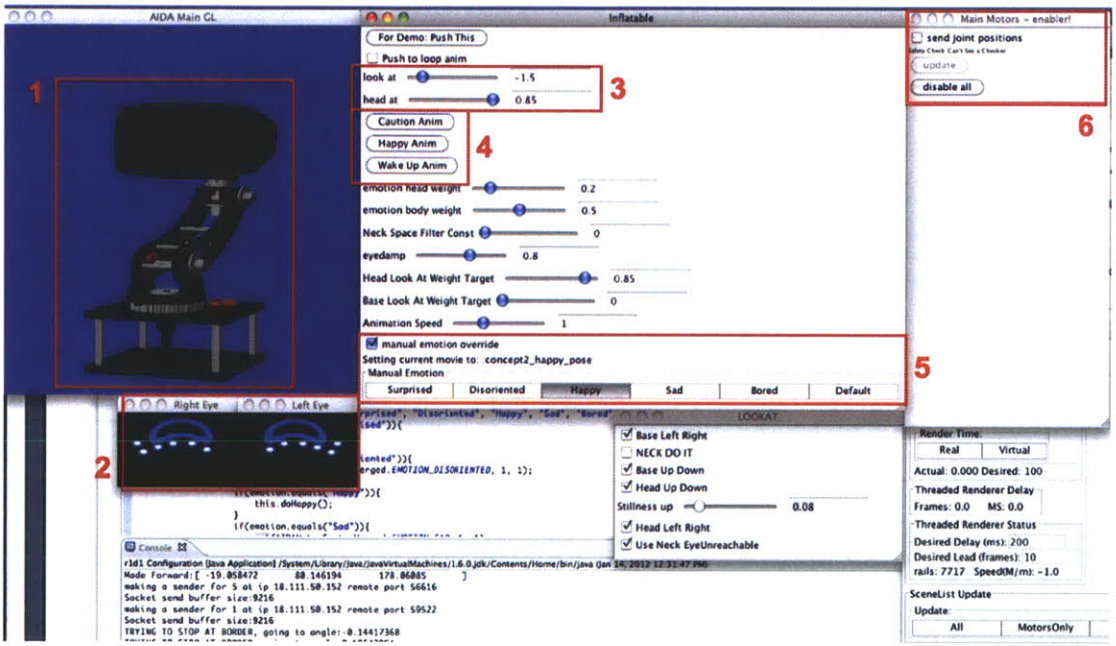


Figure 4-9: Windows of the AIDAController.
Labels: 1. 3D Animated Model. 2. Video Playback of Facial Expressions. 3. Sliders to set the Default Positions. 4. Buttons to trigger specialized Animations. 5. Buttons to set the mood manually. 6. Motor enablers.

R1D1 on The Phone

The R1D1 application on the phone serves as a communication module between the AIDA application and the R1D1 program running on the external computer. Whenever the Android Application decides what type of expression it wants to convey next, it sends a message to the R1D1 application on the phone, which must be running in the background whenever the physical robot is expected to move. As soon as this

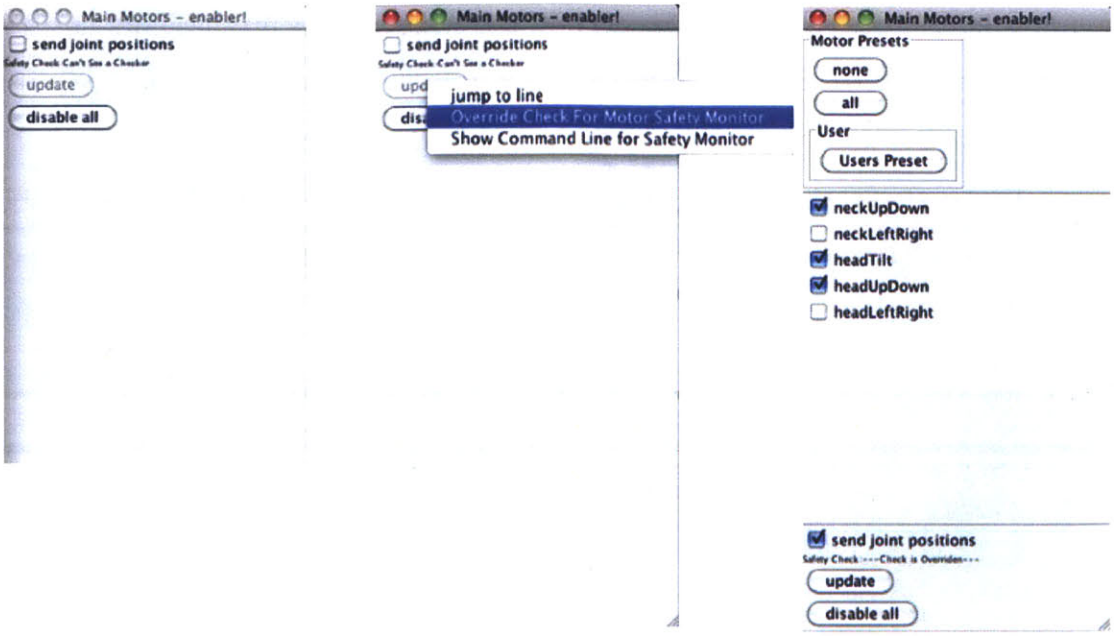


Figure 4-10: Process to enable the robot's motors

last application receives the message, it sends another message, via IRCP, to the R1D1 program running on the computer. This intermediary is necessary because the AIDA application cannot communicate directly neither with the external computer nor with the physical robot.

This program resembles the AIDAController class in the sense that it is constantly listening for commands from another layer in the system. However, since the messages are coming from another Android application that is running in the phone, the communication does not happen via IRCP, but through the transmission of Android Intents. Whenever an installed Android application wants to announce that an event has happened, it broadcasts an "Intent" with a unique string data-name. If another application has been register as a handler of packets with the broadcasted data-name, then it is notified of the message and it is given its contents.

In the system behind AIDA, the main Android application places an animation or mood request by broadcasting an Intent with a data-name that describes the desired action. The R1D1 application is registered to handle messages with those

specific names, defined in this latter class as a set of keys. Consequently, whenever a message is broadcasted with one of the key names, the R1D1 program is notified of the message, for it to take the appropriate action. It follows that it is imperative for the sender and listener to agree on the messages' names; otherwise, the commands will not be received successfully.

Once a notification is received, the next step is to send the corresponding message to the AIDAController running on the external computer. This step is achieved through an IRCP manager that makes a packet containing the integer ID that matches the requested animation as expected by the controller. If a connection can be established with the external computer, the packet is made and sent, and a success message is printed on the application screen. In case of failure, the manager keeps trying to establish a connection, while printing a message on each trial with the number of time it has attempted to send the message. However, since the R1D1 application is running on the background, these messages are not visible to the user, unless the application is brought to the front for debugging purposes. Moreover, the application has the ability of bringing itself to the front if the developer considers that there is a malfunction that needs immediate attention.

There are three main causes of failures in this intermediary stage: The first one is that the message is not successfully transmitted from the Android application to the R1D1 program running on the phone; The second explanation may be that the request received is not translated into the appropriate command for the external computer; The last reason could be a the lack of communication between this program and the one in the external computer. The cause with the highest probability of happening is the last one, either because both platforms are not connected to the same network or because the computer has a wrong IP Address for the phone. Buttons have been placed on this window to manually trigger the transmission of messages to the external computer without having to run the main AIDA application, facilitating the debugging process.

Figure 4-11 displays the Home Window of the AIDA Android phone, and what the application looks like after succeeding and after failing to send a request. These pictures were obtained by accessing the Android phone remotely from an Apple Mac computer and taking screenshots of the views. Therefore, these images have a poorer quality and a lower resolution than the application itself.

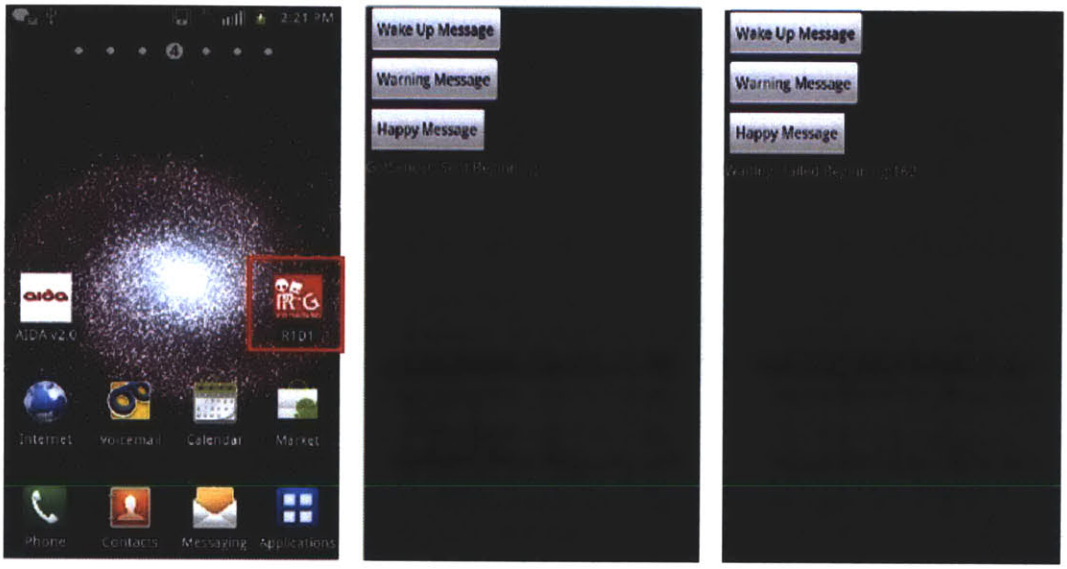


Figure 4-11: R1D1 application
Left: Application Icon on the Home Window of the Android Phone (Red Square).
Center: Debug buttons and successful "wake up" message Transmission (Text: "GotSender: Sent Beginning")
Right: application after 162 failed transmissions (Text: "Waiting: Failed Beginning: 162")

4.3.3 Android Application Overview

The Android Application is the component that is at the highest level of the AIDA system, and it is therefore the main point of interaction between the robot and the driver. Just like most of the ordinary Android Applications that are available in the market, the AIDA application lives in the user's phone and can be launched at anytime, even in the absence of a physical robot. Certainly, one of the chief functions of this application is to serve as the robot's face and behavior manager whenever it

is mounted on the dashboard. However, the users could also make use of it outside the vehicle to input additional information about their preferences, to read about the available behaviors, and to employ certain capabilities that they may deem useful, like sending text messages if they are running late or calculating their estimated travel time to their next destination. The following sections explain some of the reasoning that led to a few key design choices and it describes the framework of this application.

4.3.4 Design Choices for the Application

The current AIDA design was shaped by many choices that were made in an attempt to increase user comfort. Three crucial choices : to follow the principle of least surprise, to use speech as the main communication modality between AIDA and the driver, and to play QuickTime videos of the same facial expressions displayed on the first prototype of the system.

Avoiding Surprises

Having a moving, interactive robot sitting on the dashboard of a vehicle, has the potential to be a distracting experience just because of the innovative nature of the system itself. Whenever people are driving, it is imperative for them to stay focused on operating the vehicle, thus they should not be startled. Also, we believe that we should avoid teaching the driver how to make use of a completely new interface, because any frustration that can emerge from being unable to use the system successfully could result in stress and a higher cognitive load, compromising the safety of those inside and around the vehicle.

Therefore, it was decided that whenever possible, we would follow the principle of least surprise, avoiding any additional sources of distraction by providing interfaces that the driver is likely to be familiar with. As a direct consequence of this decision, the AIDA application tries to resemble the structure of some of the game applications programmed for Android devices; after the initial animation is played, the first

window that the user see is a menu with explicit descriptions of the available options. Additionally, there is an 'ABOUT' option with information about how the robot can be used. More information about these Activities is given in Section 4.3.5.

The interaction between the driver and the robot also tries to diminish the number of surprises. Although a conversation with another passenger can be happening constantly, even during dangerous maneuvers, we tried to limit the amount of interaction between AIDA and the driver, and the times in which it happens; It is understandable that people may need some time to become familiarized with the concept of having a robotic assistance. Keeping this in mind, AIDA is only meant to communicate messages during what we considered to be safe times, like when the vehicle is at a stop or when the car is moving at a relatively low, constant speed. Currently, this goal is partially accomplished using accelerometers, by making sure that the robot does not start an animation unless its acceleration is below a given threshold. More precise behaviors could be implemented using signals that are measured by an actual vehicle, including data from speedometers and blind-spot sensors. For the time being, this information is unavailable to the system due to our current static setup, but if the robot were to be mounted to a car, these signals could be used to expand on this safety concept.

Communication Modality

There are multiple forms of communication that could be used by AIDA to communicate with the driver, both verbally and non-verbally. Transmitting a message verbally could involve either written or spoken words. Written words would involve reading a message, which would require the drivers to take their sight out of the road and direct their attention to the phone for a couple of seconds at a time. In the driving environment, this kind of distraction would be enough to cause an accident, thus it was decided that AIDA would avoid the display of written words and it would only employ spoken words when communicating a message. Among the non-verbal forms

of interaction, we decided to focus on the display of body language and facial expressions, keeping haptic communication down to a minimum. Nonetheless, non-verbal forms would only be used as additional support for the spoken messages.

Verbal Communication Using speech as the main modality of communication between the driver and the car is an attempt to avoid degrading performance while operating the vehicle. Remaining attentive to the road requires visual processing resources, and interacting with AIDA through spoken words would mostly require auditory ones. Since the stimulus modalities are different, there should be little interference between the resources needed to accomplish each task, so the performance and safety are likely to be improved. Besides, having a dialogue with someone else in the car should be a common experience for most drivers. Thus, we are expecting that the users' familiarity with the form of interaction would help balance out the shock or distress that could be caused by the innovation of the overall system.

Another benefit of using a conversation model as a modalities of communication is that users are not forced to change between different forms of interactions. Currently, drivers can find themselves turning dials and pushing buttons in the IVI Systems, scrolling through lists and reading e-mails from a mobile phone, or typing directions and switching screens in their GPS Devices while they are driving and trying to stay attentive to the in-car warnings and signals. This situation can potentially be transformed into a simple interaction, in which drivers converse with an agent in the same way that they speak with other passengers. This agent could simply articulate responses or it could start a specific action based on a given command. For instance, the agent could send a text message to a friend whose phone number is in the driver's contact list, or it could play a song if the driver makes a request for it.

In general, the idea of having a casual conversation with AIDA may sound very appealing. However, the success behind this oral communication between a human and a robot is highly dependant on having effective speech synthesis and voice recog-

nition systems. Developing these systems is a very complex and time-consuming task. Therefore, we decided to employ those systems that already exists and have been specialized for Android devices.

The Galaxy S Epic 4G phone comes with default engines to transform text to speech and vice versa. Initially, we ran some basic experiments to test the quality of these systems and to set our standards for AIDA. These results were a determining factor when we had to decide how much interaction with the user the robot should have. To test the built-in Google Voice recognition system, we spoke a couple of sentences that were considered to be relevant and feasible commands in the driving environment, and we assessed the accuracy of the engine. In our opinion, the system had satisfactory results considering our expectations. Some of the commands used and their corresponding results are shown in Table 4.1 and Table 4.2. The variety of languages available was an attractive point, in case the user does not prefer English.

Table 4.1: Successful Results of the Voice Recognition System – 15cm away from the phone.

Spoken Command	**Result of Voice Recognition**
Could you play a song for me?	*Could you play a song for me*
Turn on the radio for me	*Turn on the radio for me*
What time is it?	*What time is it*
How late am I?	*How late am I*
Reply: I cannot make it today	*Reply I cannot make it today*
Yes	*Yes*
No	*No*

Table 4.2: Commands given to the Voice Recognition System that were at least partially unsuccessful. First Two Trials shown – 15cm away from the phone.

Spoken Command	**Result of Voice Recognition**
What's the weather like in Cambridge?	*What's the weather like in Pembroke*
	What's the weather like in Cambridge
Please call Nancy Foen	*Please call Nancy Collins*
	Call Nancy fine
What's my current estimated travel time?	*Hi Karen estimated travel time*
	What's my current estimated travel time

Some of the results were recorded accurately, exactly as they were spoken. Other commands were only partially successful, with a few of them returning the expected results on the second trial. It was then decided that except for yes/no answers, AIDA would read the words it recorded and ask the users for feedback, giving them the opportunity to correct any messages that were misunderstood. Most of the sentences that contained mistakes were those that had specific names, like city names (Cambridge) or names in the contact list (Nancy Foen). These cases could be dealt with by giving users different plausible options to choose from, but we decided to exclude these sentences from the list of possible commands given the complexity of this type of interaction. Moreover, we decided that only the predefined commands given in the 'ABOUT' Window will be accepted, each one of them having at least one unique word and one supporting word that will be used to identify the command issued. For instance, if the user asks "What time is it now?" the system will look for the words 'now' (unique) and 'time' (supporting).

On the other hand, the default speech synthesis engine did not produce the results that we were hoping for. The Epic Android phone comes with the 'Pico TTS' system installed and uses it to transform text to speech if no other engine is set up. In general, this system was good in the sense that speed settings could be customized and the sentences were spoken properly and intelligibly, rarely causing any confusion. However, even though the pronunciation tended to be good, the sound had a robotic feeling to it. Although the users may be aware that the sentences are synthesized, our preference gravitates toward richer voices that sound more like a human assistant, and less like a machine.

After reviewing a couple of text-to-speech (TTS) engines, and listening to their demonstrations of speech synthesis, we chose the 'Loquendo TTS Susan' Mobile Solution. In our opinion, the Susan voice, offered by Loquendo, is an appropriate match for the AIDA application because of its natural sounding voice and its ability to read its sentences in an expressive way by changing the intonation. Accordingly, the

sentence "Guess what", without any punctuation, would be read completely different from the question "Guess what?!" which will be read enthusiastically and with a higher pitch.

Besides, the Loquendo engine can be set in two different modes, 'messaging' and 'navigation', each one of them adds the capability to read the abbreviations that are commonly used on those contexts. The 'navigation' mode can expand the abbreviations typically used when writing addresses, like the ones included in Table 4.3. The 'messaging' mode can fully read many of the abbreviations commonly used while chatting or text messaging, including the ones in Table 4.4, and it plays sounds that correspond to some emoticons. For instance, :-) (a smiley face) is read with a laugh, while a crying sound is played for :'((a sad face) and the voice splutters when there is a :P (tongue out). Since AIDA would be handling both addresses and text messages sent to the phone, these modes would help clarify the sentences read by the robot. However, because the Susan Loquendo TTS is a paid application, we did not incorporate it in the application directly, but we use it in our prototype and we recommend it on the 'ABOUT' window.

Table 4.3: Common Navigation Abbreviations.

Abbreviation	**Expanded Word**
St.	Street
Ave.	Avenue
Apt.	Apartment
Dr.	Drive
SH	State Highway

Table 4.4: Common Messaging Abbreviations.

Abbreviation	**Expanded Word**
brb	Be right back
ttyl	Talk to you later
pcm	Please call me
lol	Laughing out loud
bbiab	Be back in a bit

Non-Verbal Communication The principal non-verbal methods of communication that are used by AIDA are: touch events, body language and facial expressions. Given that the driver is usually making use of its haptic and visual processing resources to manipulate the vehicle successfully, the interaction with this robotic agent should cause as little resource interference as possible. Therefore, there is a single interaction that requires a touch event, shown in Figure 4-12, which consists on the user simply touching AIDA anywhere in the android phone screen that is used as its face to issue a warning or a command. Given that the head should be almost static while the car is in movement, this touch event is basically the move performed by drivers to turn on the radio or to activate the hazard flashers. This requires considerably less time and attention than certain practices that are considered dangerous, like writing a text message with the phone's keyboard while driving.

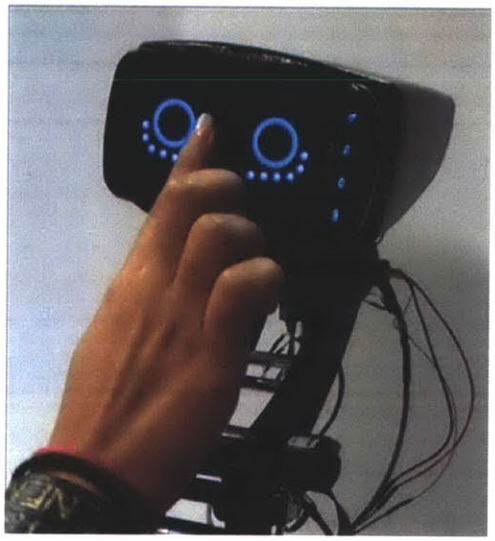


Figure 4-12: Touch action to give a warning or to activate Speech Recognition when issuing a command (depends on the setting)

Currently, touching AIDA's face could trigger one of two events, depending on whether the 'debug-mode' option is selected or not in the 'SETTINGS' Windows; if selected, touching the screen will issue one of the warnings used for debugging purposes, otherwise, it will activate voice recognition to listen to one of the preset

commands that are described in the 'ABOUT' section. The most convenient situation for the user would be for AIDA to be constantly listening for a command, using the touch event only to indicate a warning. However, the speech recognition feature consumes a significant amount of power whenever enabled, so that setup will quickly drain the phone's battery.

The other forms of non-verbal communication are used to make AIDA a more sociable entity, by giving this agent the ability to use body language and facial expressions to convey emotions. The range of moods available and the videos used to display them are discussed in the next section. As additional support for the videos displayed, the robot is supplemented with head and neck movements that are designed to improve AIDA's expressive capabilities, especially whenever it is delivering a message. Different neck extensions could be used to communicate different levels of urgency as shown by Figure 4-13, while the other degrees of freedom could make other movements more expressive and natural. Some potential scenarios could be:

- Whenever unhappy, bored or idle, AIDA will sit low on the dashboard. What is more, the feeling of sadness could be accentuated by lowering the robot's head.

- While delivering a piece of information of average priority, the robot's neck would extend to a medium height with a very slight head tilt.

- Interesting messages, or ones that have high priority could be expressed with a fully extended neck. Naturally, these movements would be avoided whenever the vehicle is moving, and would only be used when the car is stationary. Potential times include whenever the driver is getting in or out of the car or whenever the vehicle is stopped in front of a traffic light.

- Tilting the head can be used whenever the message implies curiosity or confusion. For instance, if the robot could take two different actions at a particular time, and it is unsure about what action the user prefers, a question could be

asked with a slanted head that is leaning to one of its sides, to reinforce the sense of uncertainty.

- A slow nod, conveyed by moving the head up and down, can be used if AIDA receives a request and it needs to let the user know that it understood the command and it is taking the pertinent actions.

- Turning the head or neck can be used to communicate messages to different people inside the car. If the message is meant for the driver, AIDA could face the driver seat, but if it is directed to the passenger, the robot could turn to face the passenger front-seat. At this time, all messages have to be delivered to the driver.

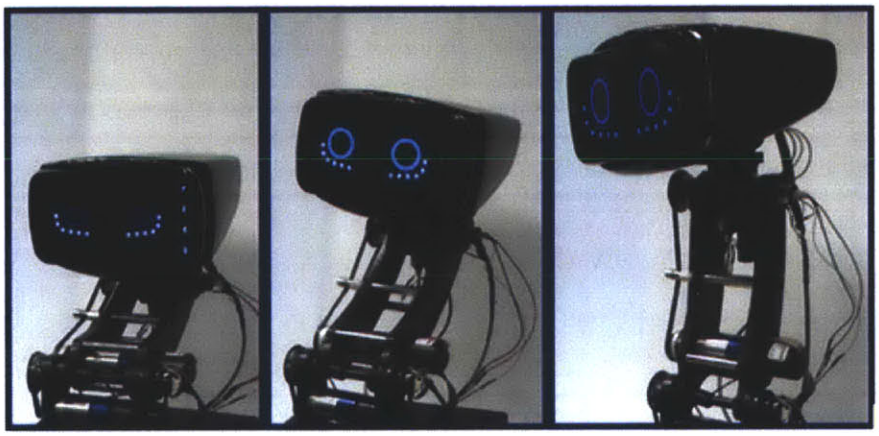


Figure 4-13: AIDA physical movements

In essence, we attempt to provide relevant information serially, through a conversation mode that feels natural and familiar to the user. All non-verbal forms of communication have been implemented to complement the message that it is being spoken, by adding some emotion to it and by offering more insight about how a particular message should be interpreted. It is our belief that this type of interaction could reduce a driver's cognitive load. Consequently, the driver would be able to focus and provide more processing resources to the driving task, which will result in significant safety benefits.

Expressions

AIDA is currently equipped with a varied set of facial expressions that can be used depending on the circumstances that surround the driver at a particular moment in time. Some of the situations in which these expressions are currently being used are:

- When greeting the people coming into the car, AIDA displays a happy face to express the idea that it is pleased to see them.

- If the vehicle is running low on gas, or if it needs an oil change, the message is delivered with a sad expression.

- After driving for an extended period of time, AIDA puts up a bored face characterized by slightly closed eyes. However, given that current user interactions are limited to a couple of minutes at a time, this expression is displayed at intervals of about 3 minutes.

- If a notification is received saying that the road ahead is congested due to a change in traffic conditions, AIDA's face will turn into a warning sign.

- Once the Android phone has received a text message, and AIDA has determined that it is safe to read it to the driver, it puts on a surprised face to alert the driver about the new text message.

There were many changes that took place throughout the development of the AIDA project, but the aspiration to make the robot more sociable through the use of facial expressions always remained constant. The variable concepts were the appearance of these expressions and the method that should be employed to display them. From the beginning, it was decided that an Android phone would be used to represent the robot's face, but there was an uncertainty as to whether the animations from the first prototype were to be reused, or if it was best to replace them.

Initially, we were inclined to create more sophisticated animations for the latest prototype. The new idea consisted on using the Maya Software to create a three-dimensional rig file that would introduce a sense of depth to the phone display. The essence of the original appearance of the face, depicted in Figure 4-14, would remain unmodified; the Maya model would be composed of two light blue hoops, which will replace the two-dimensional circles, on top of a couple of blue spheres that will take the place of the dots that were previously below the eyes.

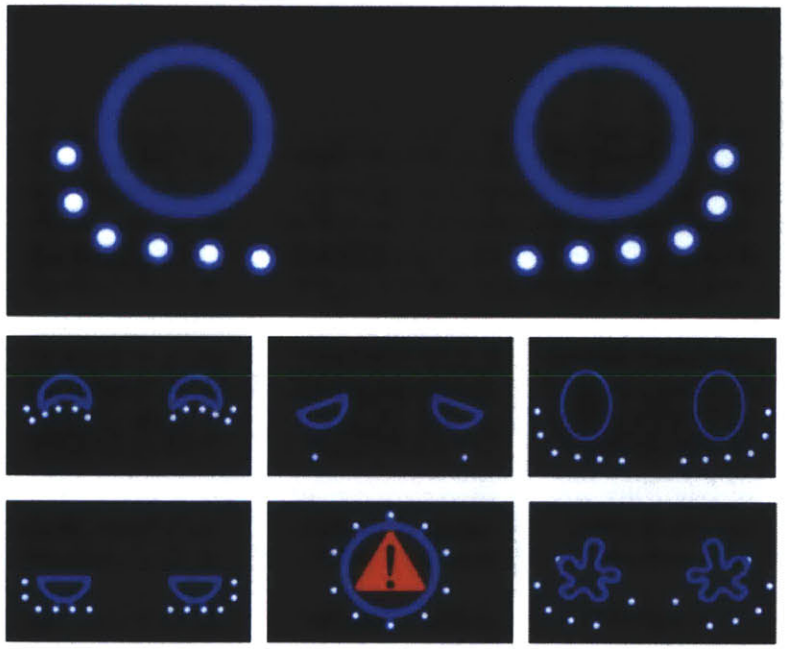


Figure 4-14: AIDA's neutral (above) and expressive faces (below). Artwork by Fardad Faridi.
Expressions shown in the middle row: happiness, sadness, surprise.
Expressions shown in the bottom row: Boredom/Frustration, Warning, Confusion

In general, the animations would be more interactive. A potential scenario would be to move one of the eyes towards the back, becoming smaller, to accompany a turn of the head, augmenting the feeling that the robot was looking to the side. Another innovative idea would be for both eyes to recess, moving back, if the user's finger was in close proximity to the face. This follows the notion that some creatures become uncomfortable when they feel like their personal space is being invaded. Moreover,

this type of file would facilitate the transition between different poses and the creation of new expressions, by allowing the developer to design new positions through simple interactions, like dragging and dropping some of the components.

The major disadvantages of using this three-dimensional model are the potential distractions it can cause and the amount of processing resources that are required to display it. Even though the user would still be able to touch the screen, the robot's reaction would now involve playing vivid, eye-catching animations that are likely to distract the driver and induce users to keep interacting with the robot. In the field of sociable robotics, this would probably be a desirable result, but under the driving context it is probably best to keep the driver focused on the road. Also, as mentioned before, we considered that given the innovative nature of the whole system, other components should be kept simple to limit how much attention the user devotes to the robot.

A more technical concern was the amount of processing resources that are required to display the three-dimensional model. Because the entire application must be running on an Android phone, the resources available to play the model are limited by the processor specifications of the phone. When a similar model for a different robot face was tested on an Epic phone, it took a significant amount of time to load. Also, its update frequency was rather low, thus the animation was not as smooth as expected, and the other processes that had to run in parallel were slowed down considerably.

As a result of the disadvantages mentioned above, and in an attempt to keep AIDA's character as consistent as possible with the first prototype's, it was decided that the same facial expressions would be reused. New short QuickTime animations were created using the previous videos. A subset of the available expressions is shown in Figure 4-14. Each new animation is meant to represent an emotion, and they all begin and finish with the neutral expression, displaying the desired emotion for a couple of seconds towards the middle of the video. This is an appropriate system to

display emotions given the way the robot always returns to the default position to transition between emotions, as described in the Motor System section.

Usually, the AIDA application triggers the video and it simultaneously sends the commands to start the physical animations. Therefore, each video was carefully created to match the head and neck movements that were designated to convey the emotion that the face is supposed to portray. Because of this, they both have the same time length, and the desired facial expression tends to be played whenever the most salient physical motion takes place. After a couple of seconds, the face returns to its neutral expression as the body returns to its default position.

At this time, the choice of what emotion to portray at each particular situation is based solely on the type of information or message that is being delivered. This ability to express emotion could also contribute to the clarity of the message and the user's understanding and appreciation of the vehicle. However, we believe that the relationship could be strengthened even further if AIDA could also perceive the driver's mood and respond accordingly. Therefore, an expansion to this project could implement a mood recognition system on AIDA and test if it enhances the quality of the user experience while improving in-car safety, as predicted by previous studies.

4.3.5 Android Application Framework

The design of the AIDA Android Application was significantly influenced by the aspiration to make the interface clear and intuitive. Consequently, the framework models the structure of other commercial applications that the user is likely to be familiar with. It particularly resembles the architecture of many game applications, and the interface is similar to the one users encounter whenever they are trying to watch a movie from a DVD (Digital Versatile Disk). Once a movie disk is inserted in a DVD player, and the user has watched a couple of trailers, there is a short video that introduces a menu screen. In this screen, there are generally several well-defined options that the viewer can select to move to a different screen.

The AIDA application, which can be installed on any Android device with an Android version 2.2 ("Froyo") and higher, also has a few components that can be accessed from the menu screen. Each one of these components is also called an "Android Activity" and it is a separate Java subclass of the 'AIDA_Activity' class. The implementation of these components was done using the Android SDK (Software Developer Kit) and the ADT (Android Development Tools) plugin for the Eclipse IDE (Integrated Development Environment).

Once the icon with the AIDA logo is selected through a screen touch, the AIDA application starts by playing a short animation. This introductory screen is called the 'Splash Activity' and it is the main Activity of the whole application. Consequently, the user is always directed to this screen whenever the application runs by the first time, or once it is restarted after quitting completely. The animation, played on a horizontal orientation, simply consists on an expansion of the AIDA logo, making it more colorful and less transparent as it grows. This project was developed in collaboration with AUDI, the automobile company, and the logo's appearance establishes this connection by being aesthetically similar to the AUDI logo. The application's version number and the full name of the project ("Affective Intelligent Driving Agent") are also specified below the logo. An image of the icon in the home screen and the Splash appearance are shown in Figure 4-15.

The introductory animation is completely unresponsive to screen touches; no actions are taken even if the user presses it or tries to drag any component of it. However, the application does react to the phone's permanent buttons, portrayed at the bottom of Figure 4-15. From left to right, these correspond to: 'menu', 'home', 'back' and 'search'. If the user presses either 'home' or 'back', the application quits and the phone returns to the home screen. Pressing the menu or the search button does not skip the animation, hence the user would only be taken to the menu screen after the logo has finished its growing and fading out animations, which take about five seconds altogether.

Figure 4-15: AIDA Application.
Left top: Icon on the home screen (red square). Left Bottom: buttons on the Epic phone. Right: Initial Animation; Text: "Your Affective Intelligent Driving Agent. Version 2.0. All rights reserved"

Menu

As suggested by its name, the 'MENU' Activity consists of a list of all of the additional activities that are available for the user to select. Each one of the four options corresponds to a different Activity that could be employed by the user to customize and make use of the entire AIDA system. After an option is selected, the phone transitions to the corresponding screen, but the user can always return to the menu screen by pressing the 'back' button. The 'menu' button has the same functionality on any of the available options except for the 'CAR MODE'.

Visually, the menu screen is divided into two sections, a relatively small image of the logo on top and a scrollable list of activities on the bottom, as illustrated in Figure 4-16. The names of the activities that compose the list are displayed with white letters, contrasting well with the black background. As the pictures show it,

all four options are visible whenever the screen is in its vertical orientation, but only the top three activities can be seen in the horizontal mode. A scrollbar appears on the side of the list whenever the user switches from the vertical to the horizontal orientation and whenever the screen is touched to be scrolled.

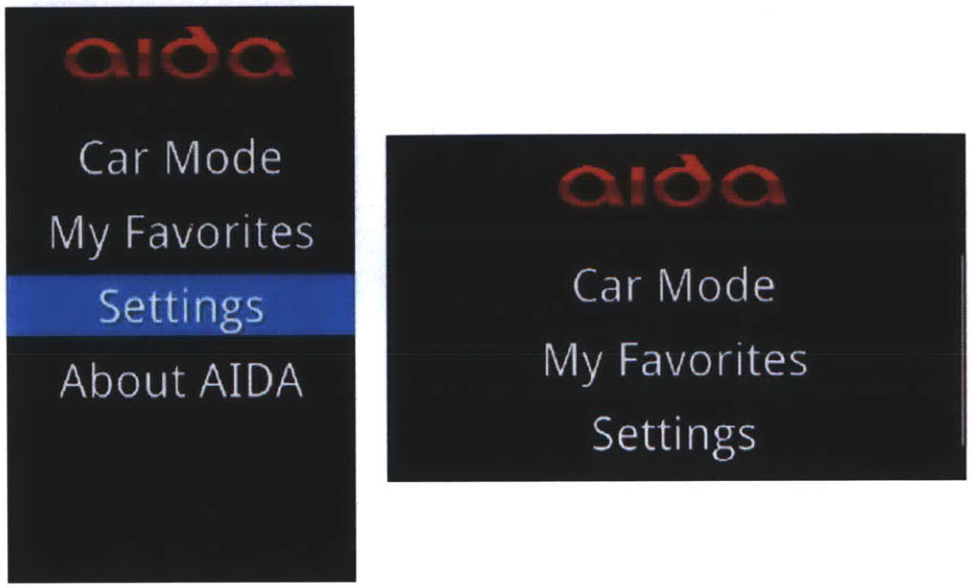


Figure 4-16: 'MENU' Screen on both orientations. On the left, the user is pressing the 'Settings' option.

If the user presses any of the options displayed, a light blue rectangle appears to indicate the item that is being selected. If the finger is lifted, the application transitions to the highlighted Activity. However, the user can keep pressing the screen and drag the finger out of the highlighted option to cancel the current selection in order to choose another alternative. Each one of the listed activities, 'CAR MODE', 'MY FAVORITES', 'SETTINGS' and 'ABOUT AIDA', are described in the following sections, in increasing order of complexity.

About AIDA

The 'ABOUT AIDA' screen corresponds to what is commonly known as the 'Help' option in many applications. However, this is an activity that strictly displays the

contents of a predefined text file, instead of providing guidance to the user based on their specific inquiries. We are aware that an application that would accurately respond to the users' particular questions would be more likely to avoid any driver's frustration when interacting with the agent. Even so, in our opinion, a general description of the available behaviors would suffice for the purposes of researching how people would respond to a sociable robot inside a car. In view of that, it was decided that this screen would be kept simple and more time would be spent into developing a higher quality interaction for the 'CAR MODE' Activity of this application.

The appearance of this screen closely resembles that of the menu. On the top, there is the small image of the AIDA logo, and on the bottom, there is the scrollable area where the white text is displayed over the black background. Figure 4-17 illustrates this screen when the phone is on the vertical and horizontal orientations, and it shows the beginning of the descriptive text on the left and the ending of it on the right.

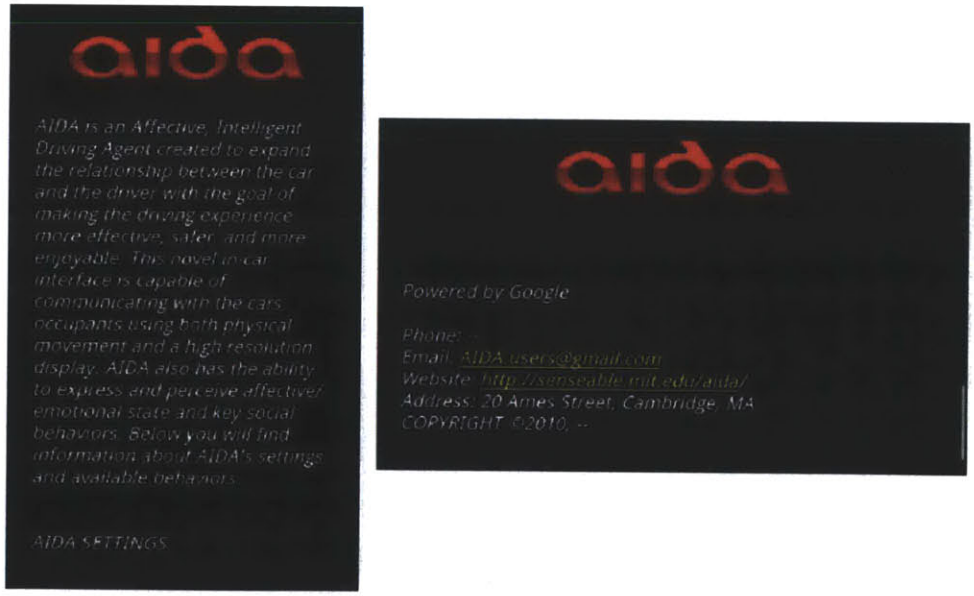


Figure 4-17: 'ABOUT AIDA' Screen on both orientation showing the beginning and end of the text from left to right.

The description given in this screen has three well-defined sections. The first one has an overview about the concepts that AIDA is meant to embody, what the overall

system consists of, and the purpose for what it was developed. Following this section, there's a description of the settings that are available, their meaning, and the effect that they have on the agent's character. Then, there is a summary of the behaviors that have been implemented in AIDA and under what circumstances they will be triggered, including a list of the commands that can be given to the robot when the voice recognition is activated.

At the very end of this screen, there is some contact information that could be used if the user has any additional questions or doubts that were not cleared by the descriptions on this window. There are two links available; clicking the e-mail address will take the user to another window where a message can be composed to be sent to that address, while the website will open a browser and load a page with additional information on AIDA.

Settings

This is the screen where users can do most of the AIDA personalization, by specifying their preferences. All information submitted in this screen is accessible by all other activities within the AIDA application and it is persistent, thus it will be available even after the application is closed and the phone is turned off. Nonetheless, none of the fields that appear in this window require any data that is necessary to run the application successfully, thus the agent will simply proceed to use its default values whenever one or more of its fields does not have any data saved in memory.

Aesthetically, the 'SETTINGS' screen was designed to match the 'MENU' and 'ABOUT AIDA' activities. In accordance with the other appearances, the logo is static on the top of the page, and the printed text has a white font over a black background. However, unlike the other windows, this screen has a couple of different fields for the user to fill out, stacked vertically underneath the logo image as shown in Figure 4-18. The 'SUBMIT' and 'CLEAR' buttons are located at the bottom of the screen.

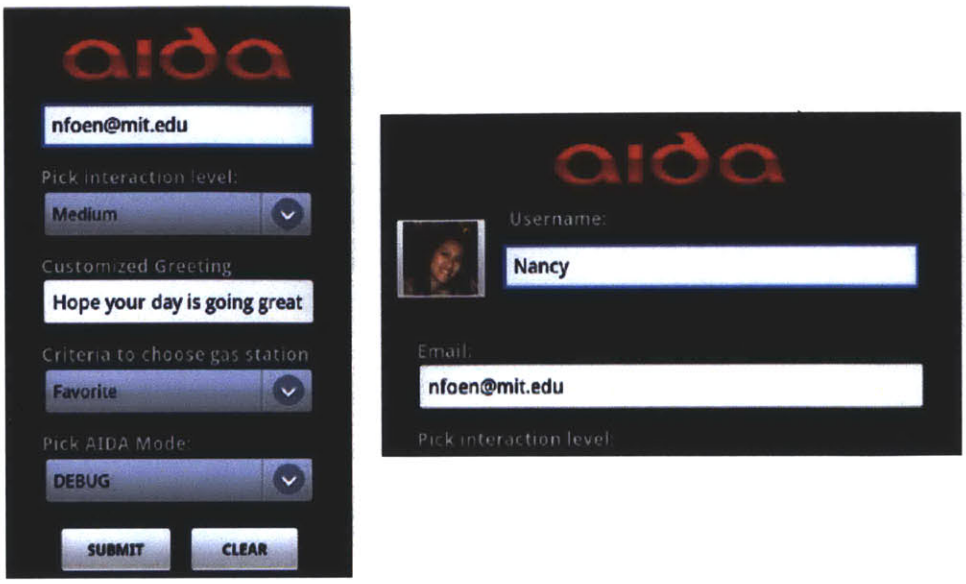


Figure 4-18: 'SETTINGS' Activity in both orientations, with full fields.

There are several types of fields in this screen, and therefore, there are different methods to input information. For most of the entries that have a string form, the application has blank TextFields in which the user can type as much information as needed in a single line. To fill in these entries, the user can use the physical keyboard or the ones that appear on the screen when the phone is closed, hiding the physical keys.

Whenever the phone is in a vertical orientation, the keyboard appears in the bottom side of the touch screen as shown in Figure 4-19. This is the most inconvenient interface given that a significant amount of precision is required to type accurately using the diminutive keys shown on the screen. The keyboard that becomes visible when the phone is in the horizontal orientation expands over the longer side of the phone, thus it offers larger keys and it makes it less troublesome to type.

Given the dangers posed by typing while driving and how uncomfortable it is to use the keyboards while the phone is mounted to the robot's head, we are expecting all settings to be filled when the driver is not operating the vehicle, using the physical keyboard, which is probably the easiest one to manipulate. Unlike a field that requests

the address of the next destination, which is particularly relevant in the driving context, the settings entries are meant to personalize the agent in general, so there should not be an urge to change them while inside the car. In the worst case, if the user tries to make changes once the phone is mounted, the keyboard offered is the largest one available for screen touch, and it is therefore the second best mode of interaction.

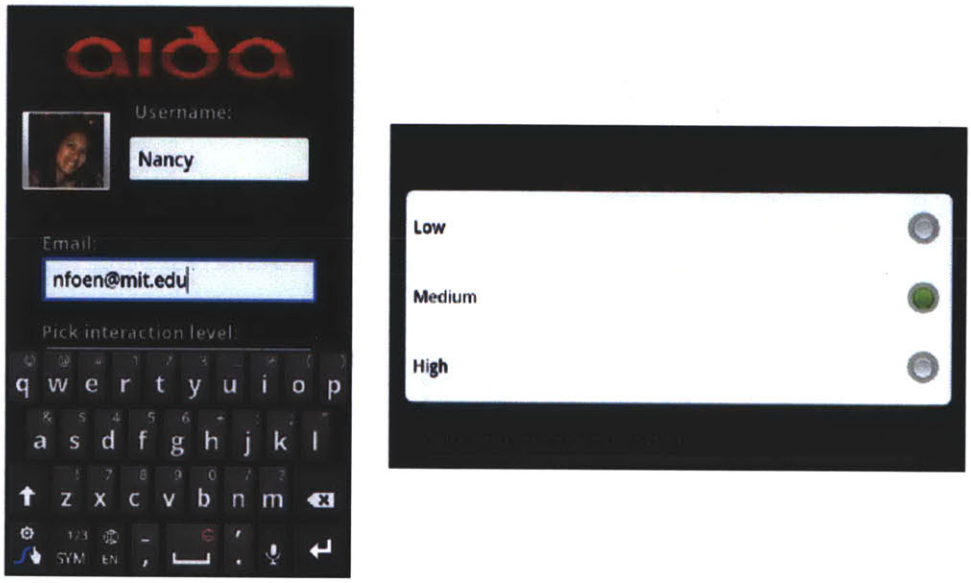


Figure 4-19: Different methods to input information in the 'SETTINGS' Activity.

An entry in which the user is asked to choose from multiple preset options is called a Spinner. This screen only has spinners that accept a single choice. A spinner field consists of a gray box labeled with the selected option, and it includes a small arrow pointing down on the right side, as shown by the 'interaction level' entry in Figure 4-18. Whenever the arrow is touched, a white window listing the available options comes into sight with radial buttons on the right of each alternative for the users to pick their preference. This window is depicted on the right in Figure 4-19.

The 'SETTINGS' screen also has a couple of buttons. The first type is an ordinary button, which simply triggers an action whenever it is pressed. The second type is an ImageButton, which has the behavior of an ordinary button, but instead of having

words on its area, it contains an image. Clicking on an ImageButton can trigger any customized action. In the Settings screen, there is only one ImageButton, and clicking it simply allows the user to select a picture from the phone's gallery to serve as button's cover. However, because the dimensions of the gallery picture do not necessarily match those of the button, there is an intermediary method that resizes the image to fit inside the button area while maintaining the original aspect ratio.

Currently, the 'SETTINGS' Activity consist of nine components:

Username In this TextField, users can type the names that AIDA should use whenever addressing them. In particular, this username is used whenever AIDA greets the driver. In the absence of a 'Customized Greeting', the default salutation would become: "Hello [username]!!! I hope your day is going well." If there were no name saved in memory, AIDA would say the same greeting, skipping the name.

Email As indicated by its name, this TextField can be filled with the user's email address. Currently, this information is not being used since AIDA can access the account that is synched with the phone. Because of this, the application does not request the password and it is not concerned with those security issues. However, the email address can be useful whenever the agent is composing a message and it wants to add it as additional contact information as plain text.

Avatar This is the only ImageButton in the Setting screen. Even though it does not have any practical function, we believe that incorporating a picture of the user in the application is likely to strengthen the bond between the user and the agent by reinforcing the sense of ownership.

Interaction Level This spinner is available to the users for them to specify how much they want to interact with the agent. There are three levels: 'low', 'medium' and 'high'. It is likely that some people would prefer a shy AIDA

that tends to wait for commands, while other would rather have an enthusiastic, proactive agent that offers its help even when the user has not asked for assistance. The default value is a medium level of interaction. This setting has not been implemented yet because at this time, we are only allowing people to have short, controlled interactions with the robot. However, the selected option is available to the 'CAR MODE' Activity and we believe that this would be an interesting path to explore in the future.

Customized Greeting It is to be expected that the users that will interact with AIDA will have different personalities. Some of them may prefer to have formal salutations, while other would like a more casual greeting. Also, because users may want to change this from time to time, to avoid getting bored with the agent, we have enabled them to customize the greeting. If blank, AIDA would use the username or the default greeting to welcome the driver.

Gas Station Criteria One of AIDA's capabilities consists on looking for gas stations. Generally, a gas station inquiry returns multiple results and sometimes, depending on the area, there could be more than ten stations to choose from. Whenever this happens, it is impractical to provide the driver with all the options found because the data becomes difficult to manage and it increases the user's cognitive load. Then, it would be useful to have a criteria to organize the alternatives in order of preference, and guess which stations are more likely to be chosen by the user. The current options are: 'closest' and 'favorite', the former being the default value. Section 6.1.4 describes this feature in more detail.

Mode Touching the screen, while the application is in 'CAR MODE' can trigger one of two actions. If the mode field has been set to 'debug', then touching the screen will open a new window to simulate a warning or an unexpected change in the environment. This is useful in the development stage when AIDA's behaviors

and responses are being tested. The other mode, 'normal', is used when a regular interaction is needed. In this latter case, touching the screen activates the Voice Recognition system for the user to be able to speak a command. There is a predefined list of commands that can be used.

Submit button After entering data, users must press the 'submit' button in order to save everything that has been inputted. Saving information implies making it available to the rest of the application, and storing it in memory for it to be remembered even after the device has been powered off. This persistent quality is achieved through the use of shared preferences, which are stored in static fields defined in the AIDA_Activity parent class. When the button is pressed, every field is set with the text that the user has inputted in the corresponding entry. If the back button is pressed without clicking the 'submit' button first, all unsaved fields will be cleared upon reopening the Settings window without saving the data.

Clear button This other button, 'clear', resets all fields and deletes any information previously saved in the phone's memory. Except for the TextFields, which become blank, all other entries will return back to their default values.

My Favorites

In this screen, users can add information about their preferences in different categories. Each category has a table in which users can add, edit and remove entries that describe their preferences. For instance, in the 'Restaurant' category, users can include the names of their favorite restaurants and what type of cuisine they serve. The information inputted in this screen is also persistent and it is available to other AIDA Activities, analogously to the settings data.

Even though the structure of this Activity consists mostly of tables, which is different from that of other windows, the appearance of the screen is similar to the

activities described above. An image of the AIDA logo stands at the top of the screen, and most of the text is written with a white font over the black background. The main component of this Activity is the set of tables in which the user's preferences are organized. Displaying all tables at the same time would require narrow columns, and small fonts for the text, probably resulting in a screen that would be difficult to understand.

Consequently, there is a tab with the tables' names and only the table that corresponds to the name selected, which is highlighted with a light blue rectangle, will appear on the screen. Similarly, given that the vertical orientation shrinks the tables and squeezes the text, it would not be aesthetically pleasing. Therefore, only the horizontal orientation is available in the 'MY FAVORITES' Activity. Figure 4-20 portrays the appearance of this activity.

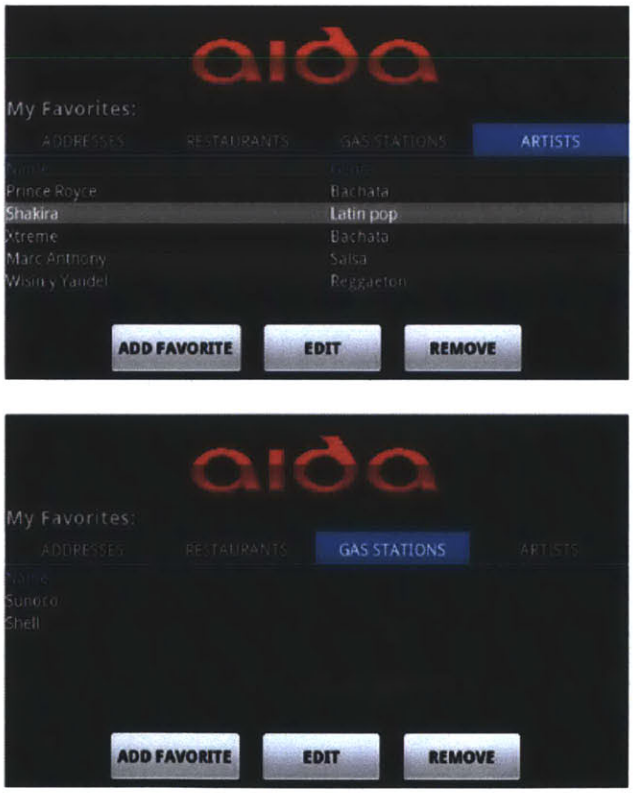


Figure 4-20: 'MY FAVORITES' Activity displaying two the tables for two different categories.

Every table in this screen is flexible, in the sense that new entries can be added and removed at any time. All changes made to the table must be submitted in order to make them persistent, keeping the users from having to fill out the tables every time the application is opened. However, because the number of 'favorite' entries varies with time, the data cannot be stored through shared preferences in the same manner as the settings. Instead, the data is kept in an XML (Extensible Markup Language) file in the phone's memory. Whenever this activity is loaded, the information in the file is used to add to the table all of the preferences that the user has inputted in the past. Rather than recording the modifications to the table while the user is interacting with the screen, the application simply uses the table entries to create a new XML file replaces the previous file. In other words, new files are created whenever the user leaves the Favorites screen.

The buttons below the tables allow the user to add, remove and make changes to the entries in each one of the categories. There are three buttons, labeled 'Add Favorite', 'Edit', and 'Remove'. As suggested by the name, pressing the 'Add Favorite' button will insert a new row in one of the tables. Since some information is needed about the entry, and about the table it will belong to, the button activates smaller windows called Dialogs that help the user input all the necessary data.

Immediately after clicking on the button, the 'Add a Favorite' Dialog appears as shown in Figure 4-21, for the user to select the desired table. The main component of this Window is a Spinner with all of the possible categories. Clicking on the small arrow inside of the spinner opens a new window similar to the one shown on the right of Figure 4-19, displaying the options. Currently, the available labels are: 'Address', 'Restaurant Name', 'Gas Station' and 'Artist'. Once a selection has been made, the user can press the 'OK' button to move on to the next Dialogue. Otherwise, the cancel button would close the window without modifying any of the tables.

The next Dialogue that comes into sight depends on the selection made on the 'Add a Favorite' Window, since it requests information that is relevant to that category.

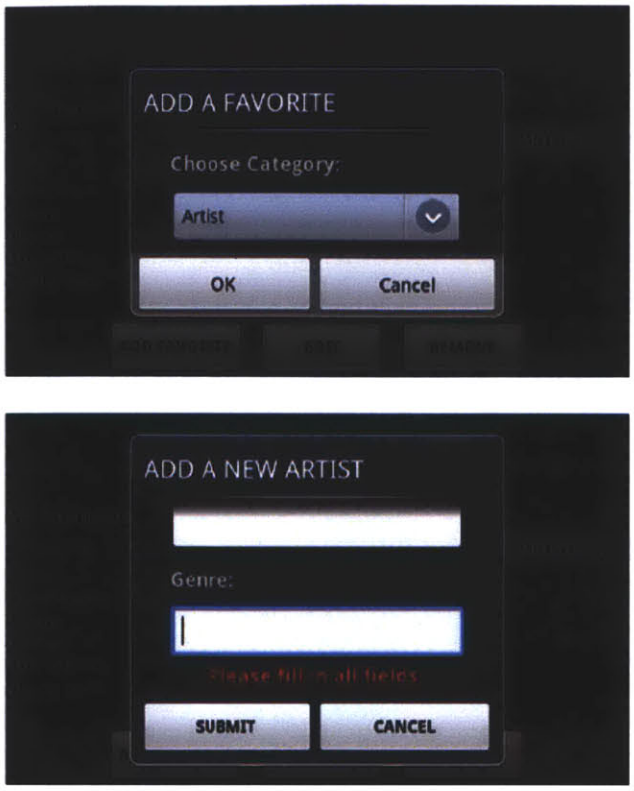


Figure 4-21: Process to add a new 'Artist' entry in the 'MY FAVORITES' Activity

For instance, if the user decides to add a new artist, the new Dialogue would have two TextFields requesting the Artist's name and his Genre, as shown in the bottom of Figure 4-21. After all fields have been filled, clicking the 'Submit' button will add the new entry to the appropriate table, and it will select that table to be brought into sight. If any of the fields is missing when the 'Submit' button is clicked, a red error code appears underneath the TextFields with the message 'Please fill in all fields' and no action is taken. Pressing the 'Cancel' button simply clears all the fields and takes the user back to the 'Add a Favorite' Dialog.

The 'Edit' button can be used to modify an existing entry. In order to do so, the user must select a row in one of the tables and then click on the button. Touching any of the rows selects it by highlighting it with a gray background, as shown in Figure 4-20. If no row is selected when the 'Edit' button is pressed, a message appears

asking the user to select the item that should be edited. Otherwise, the Add Dialog containing the TextFields appears, with the text of the selected row filled in each corresponding field. Once the changes are ready, the 'Submit' button can be pressed, which removes the original item and replaces it with a new entry labeled with the modified labels. If the 'Cancel' button is pressed, the user returns to the tables screen and no changes are made.

The last button, 'Remove', simply erases the entry selected. In the same way that the 'Edit' button needs an entry to modify, 'Remove' also needs one of the rows to be highlighted in order to delete it, and a warning message appears if none of the items has been selected. If the removal is successful, the highlighted row simply disappears from the table, and the change is reflected on the new XML file once the user leaves the Favorites screen. In general, this file becomes accessible to the rest of the AIDA Activities; if these Activities have the name of the file, they can load it and read it to extract the information needed about the user's preferences.

Car Mode

This is the main screen that drivers will use while inside their vehicles, and it is the one that displays the agent's face and activates its physical body to enable the robot-driver interaction. While at this mode, AIDA is frequently collecting data and delivering it to user as dictated by the behaviors that have been implemented in it. Data can be extracted by accessing the user's calendar, making https requests to servers like Google Maps, and inquiring about the state of the vehicle, including tire pressure, gas and oil levels; At this time, we are only simulating the gas level).

One of AIDA's main goals is to serve as a research platform in which new behaviors can be implemented and tested to assess their effectiveness and the users' response to them. The 'CAR MODE' Activity is the component of the application that is responsible for managing these behaviors and triggering them whenever it is appropriate, according to the available data. The overall flow of this Activity is illus-

trated in Figure 4-22, and the rest of this Section will describe in more detail what each of these steps entail.

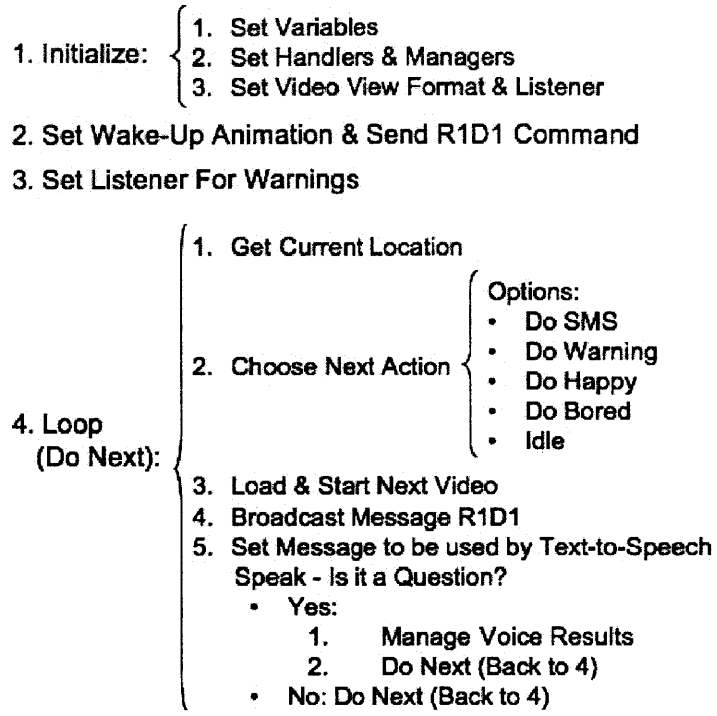


Figure 4-22: Application flow for the 'CAR MODE' Activity.

Initialization At the beginning, immediately after the user has selected the 'CAR MODE' option in the 'MENU' screen, an initialization process starts. The main function of this process is to set up the components that will be used later by other methods. Several Boolean and String variables are set up, to keep track of what modes the system is on and to store the information necessary for the agent to choose and take the next action. Also, each facial expression must be linked with a URI (Uniform Resource Identifier), for the application to know what video to load in order to express the desired emotion. Handlers and managers are other types of variables, which are instances of certain classes written to help retrieve relevant information from outside sources, including websites, the user's calendar, and the phone's contact list.

Another element that must be initialized is the VideoView component of the 'CAR MODE' Screen. During most of the AIDA's interaction with the driver, the screen should be displaying a facial expression, as shown in Figure 4-23, giving the impression that the phone is actually the face of the robot. These expressions are displayed by short QuickTime Videos, each one portraying a different emotion. Consequently, the videos must constantly be playing, one after another, simulating a continuous neutral expression that switches into the different emotions depending on the situation.

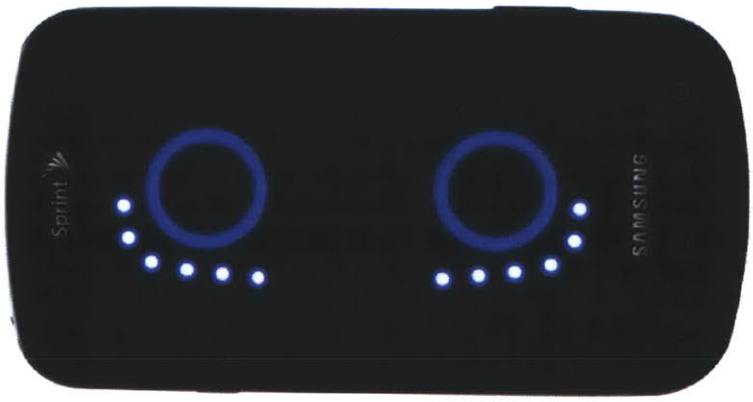


Figure 4-23: Appearance of the 'CAR MODE' Activity running on the Galaxy S Epic phone

In order to achieve a continuous effect, the application loads the next video and starts it right after the previous one is done playing. However, since loading a video takes a noticeable amount of time, the animation would flicker in between videos, making the screen turn completely black for about half a second whenever there is an emotion change. To deal with this problem, a still image of a neutral expression covers the screen during the time of the transition, and because all videos start and finish with a neutral look, this strategy makes the animation seem uninterrupted.

The speech synthesis and voice recognition systems must also be initialized when the 'CAR MODE' Activity runs. AIDA uses the default engines that have been installed on the phone, and have been set by the user in the phone's settings, which are different from the application 'SETTINGS'. Moreover, the greeting that the agent

says at the beginning needs to be defined. The default sentence consists of a general greeting, resembling an assistant's welcome in the mornings. If the driver has specified a 'Username', AIDA incorporates that name in its greeting, giving a more personalized feel to the agent. What is more, if a 'Customized Greeting' has been set, the application repeats that sentence exactly, completely overriding the default values.

Following the speech engines setup, the activity proceeds to start the initial animation. The first step here is to broadcast an "Intent", with the name "beginning_AIDA" that makes a request for R1D1 to trigger the wake-up animation on the physical robot. Then, the corresponding video is loaded, and after a delay of a couple of seconds, the video is played. The delay was introduced to establish a better match between the body movements and the facial expressions, such that the opening of the eyes are shown right before the neck begins to extend, and the surprised face is displayed when the robot's head is at its highest. Similarly, the message defined while setting the text-to-speech system is also delivered in harmony with the videos. Because of this, the greeting begins exactly when AIDA's portrays the surprised look.

Another issue that needs to be taken into consideration is the fact that in some cases, unexpected changes must activate certain actions. If the 'Debug' mode is enabled, then some of those changes can be simulated with a screen touch. On the other hand, if the application is in 'Normal' mode, then the Voice Recognition system must be activated for users to make a request by saying a command. This feature was put in place because even though AIDA is highly proactive, it cannot predict all of the user's needs. In either case, a Listener for the VideoView component must be put in place to trigger the appropriate responses after the user touches the screen.

Besides the user, other entities can also request an action; for instance, according to one of the implemented behaviors AIDA should inform the driver when a text message is received. All changes that demand a response are considered Warnings by the system. Regardless of the source of the warning, the immediate reaction is always

the same, and it consists on setting the appropriate Booleans and Strings that will be managed by the Activity the next time it decides what action should be taken next.

Continuous Loop Once the activity finishes its initialization process, it enters a loop that is continuously gathering information, and using it to decide what the agent should do next and how to do the selected action. This loop is started whenever an animation or behavior is complete, and it always starts by using the GPS services or Wireless networks to calculate the phone's current location. This information is not immediately used, but it becomes available for any of the implement behaviors to make us of it.

Then, the application proceeds to determine what action should be taken next. Generally, whenever new data is received or collected, the method that is responsible for handling it manipulates a set of Booleans, indicating that the information received needs the driver's attention. Whenever the activity is deciding on the next action, it checks these Booleans in the following order of priority: Pending SMS, Warning Messages, Happy Messages, Bored Animation and Remaining Idle. Currently, the first three modes are a result of the behaviors implemented, explained in Chapter 5. The last two are constant regardless of what reactions are being tested, and they merely display the corresponding animations. The bored emotion is displayed if too much time has passed since the last interaction with the user, while the Idle mode takes place if none of the other options have been triggered. The last case incorporates a blinking animation to augment AIDA's natural behavior.

After the action and corresponding emotion has been decided, the method that corresponds to that decision is called. Even though the information delivered by each method is different, they all have a similar structure. Each mode is linked to a different video, but it is imperative for all methods to load and start the next video that has to be displayed. This is basically the only requirement that must be satisfied by all cases. An optional task is to broadcast an Intent letting the R1D1

application know that it must request a particular set of body movements. This step is not essential given that the robot is always supposed to be moving its body in an idle mode, and in most cases it would not be unnatural for it to simply deliver the necessary information without any special movements.

Another possible action is to define the message that must be given to the driver and activate the text-to-speech engine for AIDA to say it. This message needs to be in one of two formats; it must either be a regular sentence or a question. If it is the former, then the information is merely delivered, and then the activity would return to the beginning of the loop. Otherwise, the Voice Recognition engine must be activated, as shown in Figure 4-24, and an answer is expected from the user. A manager handles the response recorded and triggers the pertinent actions. Depending on the user's commands, the manager can establish a controlled conversation between the robot and AIDA. However, the interaction is limited, and the activity always returns to the beginning of the loop once the exchange of information has finished.

Figure 4-24: 'CAR MODE' Activity when the Voice Recognition is activated.

In essence, the 'CAR MODE' Activity is the established framework in which new behaviors can be incorporated and assessed. Its general flow is rather systematic and flexible, thus a brand new action may be implemented by adding all the necessary components, which include: a triggering condition, a method to test such condition, a few variables to keep track of its state, a video and a corresponding URI for the

phone to display, an additional statement in the process that decides the next action, and a method to be called if the condition is met. It is then up to the programmer to decide how sophisticated or how simple each one of these components should be.

In our opinion, the four options that are currently available in the 'MENU' screen are enough to demonstrate AIDA's basic functions and to test what people's responses are to this innovative research platform. In the car mode, the scenarios developed depict the interaction between the different applications, and they demonstrate how the information collected is delivered to the driver. Further expansion is possible by incorporating additional options, more applications to interact with and alternative behaviors to respond to the collected data.

Chapter 5

Implemented Behaviors

5.1 Purpose

AIDA is meant to be a friendly driving assistant, equipped with a set of behaviors designed to support the user in a wide range of situations. Ideally, a reaction will be included to respond to almost any circumstance that can take place in the driving environment. However, because this is a research project and not a commercial product we have developed specific scenarios in order to test the general concept behind this agent and determine what the users' true values and needs are. Once an evaluation takes place, the users' reactions will give us a sense of what is expected of this interface, what behaviors have higher value, how intuitive it is to use, and whether or not it actually improves safety and the quality of the user experience.

5.2 Target Scenarios

5.2.1 Seamless Transition

Overview

Often, people use multiple applications and sometimes they even use more than one device when trying to accomplish a task. For instance, a user may add an event to a calendar with a specific start time and location. Later, when the user gets into a vehicle, the location must be inputted into a GPS device to get routing directions. However, if the location is not remembered, the user has to access the calendar through a mobile device before they can type it in the GPS device.

AIDA offers an alternative to this multi-step process; a user can simply enter a new event to their calendars through a browser in their computer, while still outside the vehicle, and have AIDA automatically know about it and act accordingly. The calendar update can happen either with days in advance or right before the user gets into the car. Once inside the vehicle, the user snaps the android phone in the front area of AIDA's head and, assuming the calendar is synced with their android's phone

calendar, AIDA will wake up and it will automatically be aware of the driver's next event. Then, with a happy mood, the agent will provide relevant information to the driver, including the name of the event and the estimated travel time.

Implementation

In order for this behavior to be successful, it must deliver the information immediately after the 'CAR MODE' Activity is selected. Otherwise, the driver will be more inclined to start manipulating the applications necessary to extract the information without the agent's assistance. In view of this, the message is provided during the wake up animation, at the beginning of the activity. This means that the necessary information must be gathered as the application is loading.

The main challenge behind the collection of data lies in the fact that multiple sources must be used to build the desired message, as shown in Figure 5-1. The first step consists on extracting all the pertinent information about the driver's next event, as indicated by the phone's calendar, which is supposed to be synched to any outside calendars that the user deems important. As a result of this step, the application gains information about the next event's title, location, time and participants.

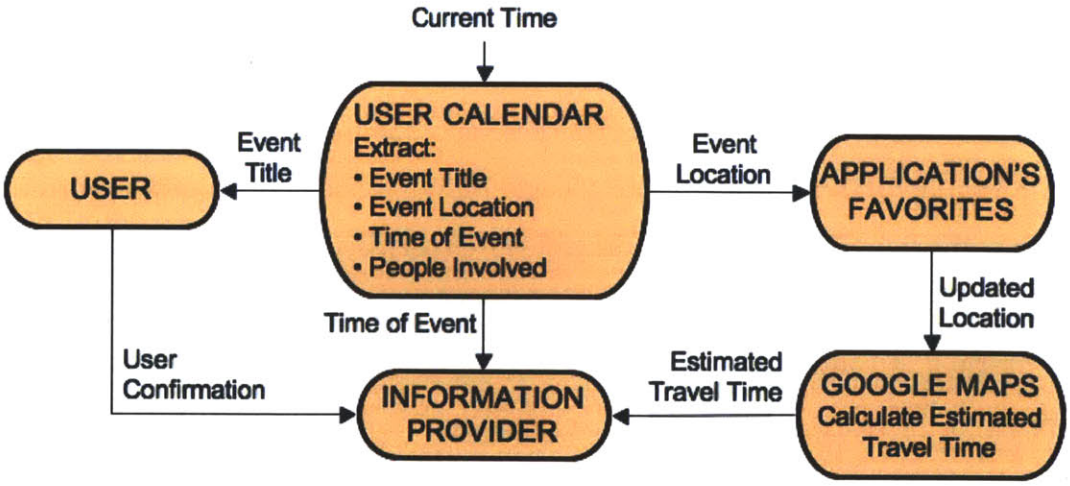


Figure 5-1: Process used by AIDA to provide information about the driver's next event.

The event title is used to confirm the validity of the event with the user, since there is a possibility that the user is not planning to attend the next event marked in the calendar. The confirmation question is asked with the text-to-speech engine, which activates the Voice Recognition System and the Result Manager to process the driver's response. If the user says that the event is incorrect, the interaction finishes with AIDA saying: "OK then, please let me know when I can be of service". If not, the behavior continues its course.

At the same time the agent is confirming the event, it attempts to calculate the user's estimated travel time. Since the application also accepts the names of one of the favorite addresses, inputted in the 'MY FAVORITES' Activity, a location like "Jane's house" is allowed. However, this description will not be recognized as an address by a map application. Because of this, the application must check if the event location is matched to an actual postal address in the Favorites XML file, using an Address_Handler. If this is the case, the event location is updated using the information given by the table entry. At the end of this step, the location must either be a postal address or the name of well-known point of interest.

Then, AIDA will attempt to calculate the estimated travel time to the destination using a Geocoder and an https request to Google Maps. The Geocoder is an Android class that transforms location descriptions into geographical coordinates and vice versa. This project uses it to extract the latitude and longitude of the location obtained in the previous step, because these coordinates tend to produce less errors than postal addresses when fed into Google Maps. If the http request still results in an error, the behavior is skipped altogether. Otherwise, a JSON (JavaScript Object Notation) Object will be returned, and it will be used to calculate the total travel time of the first route suggested.

Once all of the information becomes available, the agent must deliver it to the user using the speech synthesis system again. The spoken message includes the event scheduled time and the estimated time of arrival, as in the following sample response:

"Your estimated arrival time is 1:32PM and your next event starts at next 1:40PM". This sentence completes this behavior.

In essence, this scenario explores the concepts of having a centralized source of information and a proactive driving assistant that is available everywhere.

5.2.2 Unexpected Changes

Overview

This scenario was developed as an extension to the 'Seamless Transition' behavior described above, because even when drivers plan ahead of time, and leave their starting location with plenty of time to arrive early, the situation may change due to traffic or weather conditions. In this scenario, AIDA tries to keep the user from doing additional tasks as a result of unexpected changes in the surroundings, thus trying to lower the cognitive load.

As the user is driving, AIDA frequently recalculates the estimated travel time to make sure that the driver is on time. If one of these recalculations says otherwise, AIDA immediately estimates how many minutes late the driver will be and, assuming that it has the necessary contact information available, it offers to send a text message to the people involved to let them know about the driver's tardiness. If the user approves, a message is composed and AIDA tries to send it, reporting a success notice or a failure explanation back to the user depending on whether or not the message was sent successfully.

Implementation

Ideally, AIDA would be constantly recalculating to driver's location to verify if there has been any significant change in the estimated travel time. However, given that the current environment is static, and that the traffic conditions cannot be manipulated to provoke a delay, it is rather difficult to simulate a situation in which the driver is

running late. Because of this, a screen touch is used in the 'DEBUG' mode to indicate a change in the travel time. Pressing on the phone screen opens up the window shown in Figure 5-3, and clicking the 'OK' button when the option 'Late Warning' is selected would activate this behavior.

Figure 5-2: Simulating a Late Warning in the 'CAR MODE' Activity.

Once the warning is received, the estimated travel time increases by a set amount of minutes (currently 45min), and the process depicted in Figure 30 is activated. Given that a set delay is not always enough to make the driver late, there is a tardiness calculator that uses the time of the event, and the updated estimated travel time to determine whether or not the change is relevant to the driver. Making this decision involves approximating the new arrival time and computing the difference between this and the event time. If the driver is more than five minutes late, AIDA immediately offers to send a text message to the people involved in this event, specifying how late the driver will be.

The actual delay is rounded to the nearest multiple of five, thus if the driver will be 13 minutes late, AIDA would ask: "Would you like me to send a text letting [other participants] know you are going to be 15 minutes late?" Then, the Voice Recognition and Result Manager systems are activated to process the user response. If the user refuses, the interaction finishes in the same way as the previous behavior, by having AIDA says: "OK then, please let me know when I can be of service". Otherwise, the

confirmation is transmitted to a sender that is responsible for composing and sending SMS messages to those who may be waiting for the driver.

The SMS Sender receives the estimated delay from the tardiness calculator and the confirmation from the user, but in order to send the required messages, it also needs the contact information of the message's recipients. Assuming that the calendar description has the contacts' name, like in "Meeting with: John Smith", AIDA looks up the contacts' information in the driver's contact list, and if found, it will extract all data related to those entries, including their phone numbers. Assuming that the driver has agreed to send the text, AIDA attempts to dispatch the message to the numbers found in the contact list. Then, the application listens to broadcasts made by the phone's messaging service, to detect if there were any errors during the transmission. If the result is positive, the agent happily informs the user about the delivery success, but if an error occurred, it apologizes and it provides the error message.

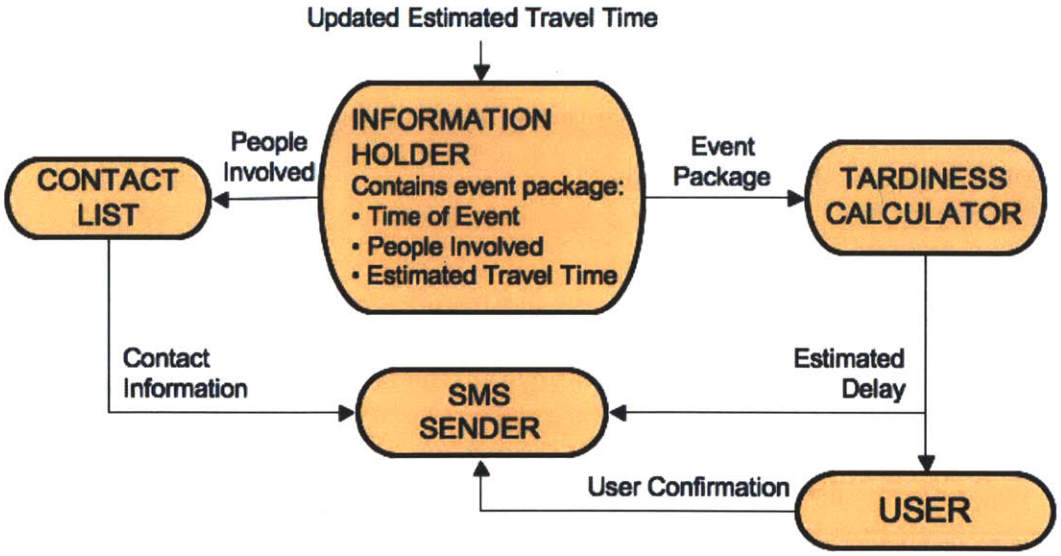


Figure 5-3: AIDA's response when a driver is running late.
Note: the boxes do not symbolize a Java class; they represent an entity or a collection of methods and applications used to accomplish what their labels are describing.

Except for the initial late warning, all communication between driver and agent in this behavior is done through speech. In this scenario we explore the concepts of

improving in-car safety by keeping the mobile phone away from the driving hands, and having a driving assistant that reacts to unexpected changes. The objective is to keep stress away from the driver as much as possible by avoiding additional tasks.

5.2.3 Receiving a Text Message

Overview

In the previous scenario, an action was implemented to proactively compose and send a text message to reduce the times in which the driver has to manipulate the phone while inside the vehicle. Typing and sending text messages while driving is known to be a dangerous practice, as the drivers' attention and processing resources deviate from their main task. However, receiving SMS can also threaten people's safety as drivers feels an immediate urge to read the text, thus taking their eyes off the road for significant periods of time. Therefore, we have also developed a relatively simple behavior to handle incoming text messages. When a message is received, AIDA simply offers to read it to the driver at a suitable time and if the driver agrees, the contents are communicated through speech.

Implementation

This scenario is similar to the ones above in the sense that most of the necessary information is stored in some of the application's variables, and other applications and methods are used to collect any additional information that is necessary. At the end, with the user's permission, the system provides a relevant message through speech. The flow of information in this behavior is shown in Figure 5-4. The condition that triggers this reaction is a broadcast that provides information about a new SMS Message. A subclass of a BroadcastReceiver is constantly listening for these broadcasts to let the AIDA application know whenever there is a message that requires attention.

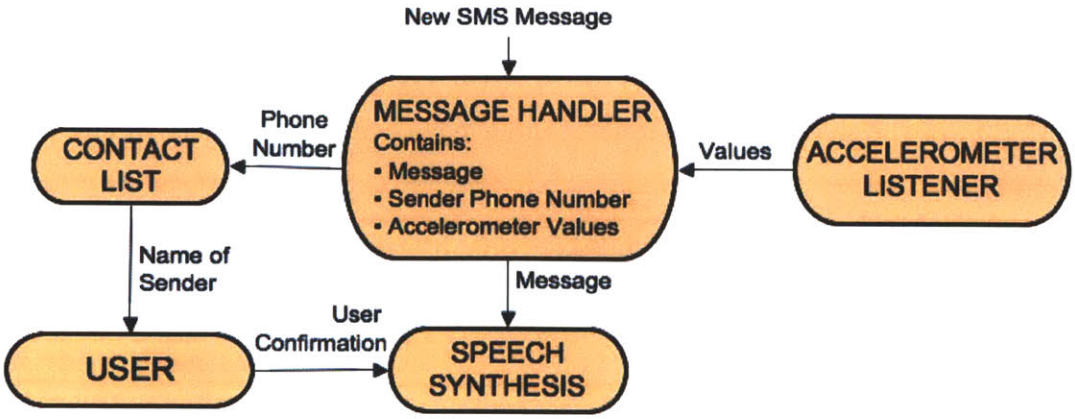


Figure 5-4: AIDA's response when receiving an SMS message.

Even though the messages should be delivered to the driver as soon as possible given that the contents may be time-sensitive, AIDA still needs to determine what is an appropriate time for this interaction. In our opinion, it would be better to have a delay of a couple of seconds, rather than having the robot rise and deliver the message during a risky maneuver. In an optimal scenario, the agent would have access to the vehicle's lane's switching sensors or even sensors on the turning wheel to recognize dangerous moves, but since it is not integrated in a car, other alternatives had to be explored.

It was then decided that the application would make use of the android phone's accelerometers, even though they cannot be tested on a stationary simulation environment. Nevertheless, we wanted to implement a feature that would stand for the concept that AIDA was context-aware in its attempts to increase user safety. For testing purposes, we could use the software that will be used in the simulation environment send signals mocking the sensors' readings.

Accordingly, the accelerometer sensor is initialized at the beginning of the 'CAR MODE' Activity and a listener is established to update a variable containing the data reported. The sensor readings have acceleration values in three directions; when the phone is in the position portrayed in Figure 4-24 these directions correspond to moving the device upside/down, side/side, and out/into the page. Given that this

will be the phone orientation in the vehicle, only the last two values were considered when determining if it was an appropriate time. More specifically, the message is not reported until the side/side acceleration and the change in front/back values go below certain thresholds.

As the application waits for a suitable time, it also uses the contact list to replace the phone number of the message's sender with a proper name. If the number is not recognized, it is kept unchanged. As a result of this, the application is ready to ask for the user confirmation as soon as the accelerometers indicate that it is a suitable time. The robot then says: "NEW MESSAGE!!! You have received a message from [sender name/number]. Would you like me to read it?" If the driver refuses AIDA simply finishes the interaction by announcing that it will be available whenever the user may need it. Otherwise, AIDA proceeds to read the text message contents; if the loquendo engine is used, it will expand abbreviations and it will make sounds that correspond to the smiley faces.

Basically, this scenario evaluates AIDA's effectiveness in increasing safety and providing user support. If tested, it would provide a good measure of how comfortable users feel about not having their mobile phone on their hands. A possible expansion would be to include default responses, similar to "I am driving, I will call you back later", or asking the user to dictate a reply.

5.2.4 Low Gas Warning

Overview

This last behavior revolves around the difficulties that may occur when a vehicle is running low on gas, and it is highly independent from the scenarios described in the previous sections. Often, drivers decide to ignore the low gas warning as it goes on, instead of stopping at a gas station as soon as one comes into sight. After that, it is common for them to get used to the lit signal, overlooking it until the fuel tank is almost empty. When this happens, users must absolutely go to the nearest gas

station, even if the gas prices are relatively high at that particular location, and regardless of whether or not they are late for their next event.

Motivated by this scenario, we have implemented a behavior in which AIDA reacts to a low gas warning by searching for gas stations that may be convenient for the driver, and suggesting them in a conversational manner. The following information is used to assess the available stations: current and next event locations, time until next event, fuel available, average mpg (miles per gallon) and the user's setting for 'criteria to choose gas station'.

Implementation

The amount of information that must be taken into consideration in this scenario makes this behavior moderately more complicated than the ones described before. The general structure is carefully explained in this section and the flow of information is depicted in Figure 5-5. Because AIDA has not been integrated to an actual vehicle, a screen touch warning triggers this behavior, opening the window shown in Figure 5-3 whenever the application's mode is set to be 'DEBUG'. If the 'Fuel Warning' option is selected, the application transitions to another window that asks the user to provide the mpg and the number of fuel gallons left in the vehicle.

Once this information is submitted, AIDA starts to gather the data necessary to obtain a list of possible gas stations where the driver can stop at. The first step in the process is to determine whether or not the vehicle has enough gas to arrive to its destination, which is registered in the phone's calendar, and then make it safely to the closest gas station around there.

In the previous scenarios, the application extracted the driver's next calendar event, together with its start time and location. Using this information, the agent can calculate the total distance that the vehicle must travel to arrive to its destination, by making an https request to Google maps. Then, using the vehicle's mpg (miles per gallon) and remaining fuel, the application can determine how many miles the car can

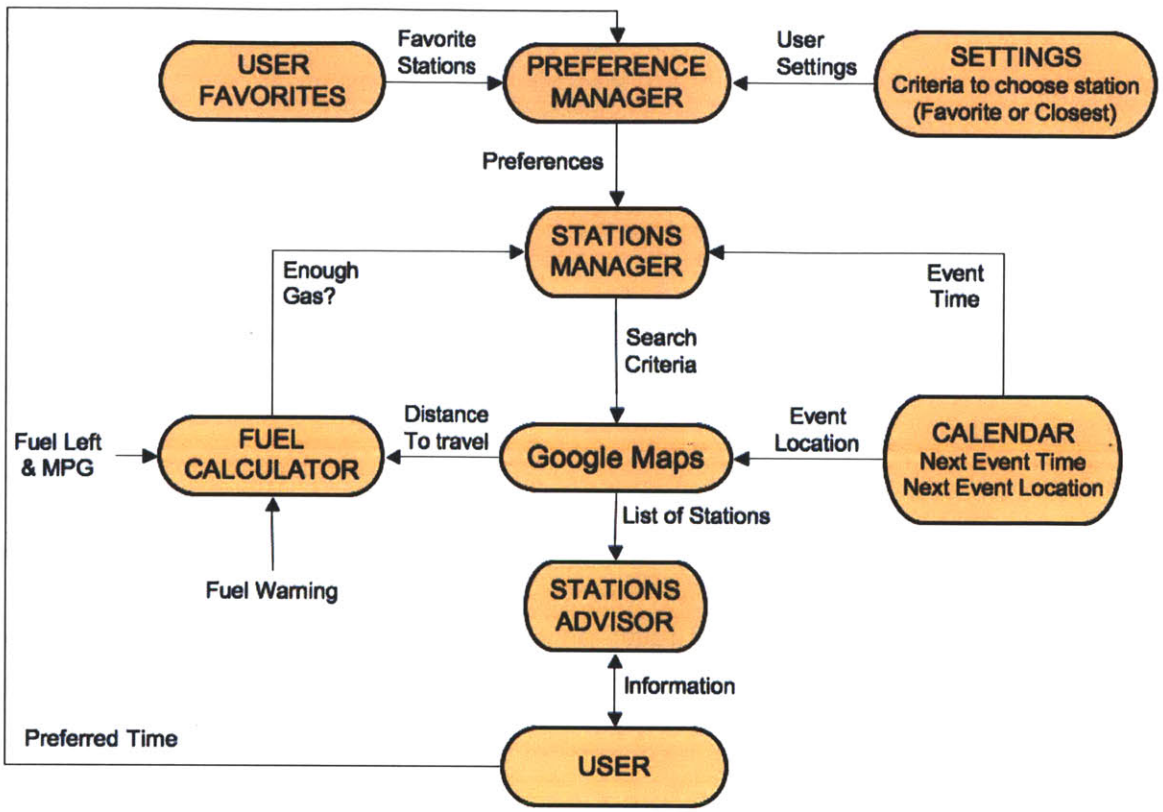


Figure 5-5: AIDA's reaction to a Low Gas Warning.

travel without exhausting all of its fuel. Subtracting the distance to the destination from the total distance that can be traveled will provide a result that reflects the maximum distance that the driver can travel safely after reaching the event location. If the distance to the nearest station within this radius is less than the miles that the vehicle has left, then the fuel is indeed enough.

Figure 5-6 illustrates the decision making process behind this scenario. From the diagram we can deduce that if there is not enough fuel, the driver is forced to look for a gas station near the vehicle's current location. However, if this is not the case, the next step is to figure out whether or not the driver has enough time to go to the gas station before the next scheduled event. This decision comes from the fact that even if the vehicle has sufficient fuel, drivers may have some time before their next event to fill up the gas tank, so that they do not have to worry about it in the future.

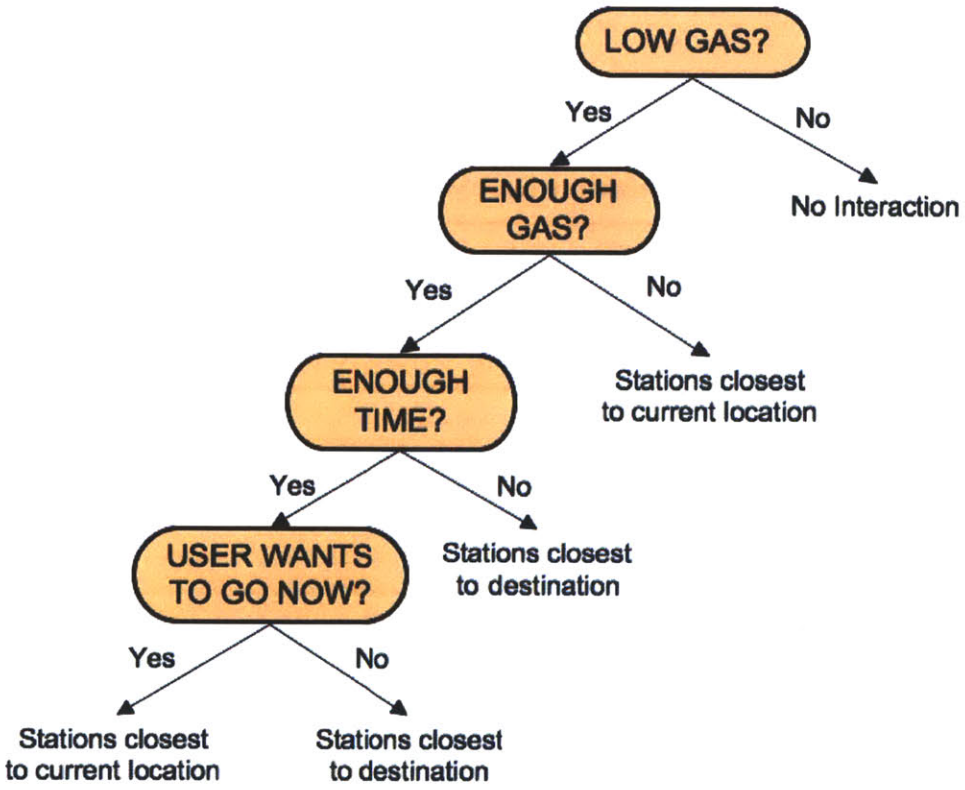


Figure 5-6: Process to determine the Search Criteria.

The first scenario, described in Section 5.2.1, implements a method to calculate the estimated travel time. In this behavior, the agent estimates what would be the travel time if the user drives by a gas station on its way to the event location. Then, a new arrival time is calculated using the updated travel time, the current time, and an additional delay of 10 minutes, which is the approximately the amount of time the user will take to fill up the gas tank. If the new arrival time happens before the event start time, the application concludes that the driver has enough time to stop for gas. If this is the case, AIDA proceeds to the next step shown in Figure 5-6. Otherwise, if the user does not have enough time, the agent would try to look for stations around the destination and it would suggest those instead.

Assuming there are enough fuel and sufficient time, the next step is to ask the user if there is preference to stop at a station as soon as possible or not. If the user denies

it, then AIDA would look for stations around the destination, but if there were a preference to go immediately, it would search for stations nearby its current location.

After the decision has been made as to where to look for stations, and assuming that the criteria set by the user in the 'SETTINGS' Activity was to look for the closest stations, an https request would be made to Google maps requesting all gas stations around the desired area. However, if the criterion is to look for favorites, an additional parameter is included in the https request, specifying the names of the user's preferred stations in order for Google to return filtered results. Once the list of stations is obtained, the potential messages from AIDA would be:

NOT ENOUGH FUEL "You are running low on gas. I don't think you have enough fuel to go to your destination before going to a gas station. You could stop at the [nearest station name] by [nearest station vicinity]. Would you like another option?"

NOT ENOUGH TIME, BUT SUFFICIENT FUEL "You are running low on gas, but you have enough fuel to go to your next event before stopping at a gas station. You could stop at the [near destination station name] by [near destination station vicinity]. Would you like another option?"

ENOUGH TIME AND SUFFICIENT FUEL "You are running low on gas, and you have some time to stop by a gas station now, before the [event title]. You could go to the [nearest station name] by [nearest station vicinity]. Would you rather go later?"

Each one of the cases above ends the sentence with a question. If another option is offered, and the driver says 'yes', then AIDA proceeds to the next station on the list, and it asks again if another option is preferred, repeating this interaction until the user is satisfied with the option or until there are no more items available. On the other hand, if the driver says 'no', the application then reads the exact location of the station agreed on. In the last case, if the user prefers to go later, AIDA goes

into a similar interaction offering the stations close to the destination, until the user agrees to go to one or until no more options are available.

A possible improvement to this behavior is to search for the most cost effective station in the area and offer these options to the driver. Anyways, the described scenario explores the idea of having an assistant that knows about the vehicle, the driver and their surroundings. It also tests the ability of the agent to offer what it believes is the most suitable information at the most appropriate time, considering the driver's preferences and the priority level of the information.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

AIDA, an Affective Intelligent Driving Agent, has been developed as an extensible research platform designed to explore new ways to improve in-car safety, while enhancing the quality of the user experience. AIDA is a highly expressive robot that sits at the dashboard of a car and behaves as a proactive assistant, attempting to reduce cognitive load by helping the driver perform certain tasks, like managing an agenda, e-mail and SMS, entertainment system and vehicle warnings. AIDA's sociable nature was tailored to explore the emotional bond between driver and car. By showing concern for the driver, and a disposition to help, users can feel a greater sense of attachment to their vehicles.

While it is true that his project could pose a distraction concern, we believe that a conversation mode of interaction with an expressive, sociable character should feel natural to the driver. If this is the case, we anticipate that AIDA is not going to be significantly more distracting than another passenger. Also, since the main computational unit is an Android phone, this concept keeps the phone out of the driver's hands, avoiding further interference with the driving task.

Currently, AIDA is equipped with several social behaviors that are triggered at specific situations, like when a text message is received, or when the driver is running late for an event. Even though it would be ideal to have an intelligent, reliable agent that could be useful under any given circumstances, our immediate objective is to determine whether or not AIDA is an appropriate interface for the driver to communicate with the vehicle, other applications and their surroundings.

6.2 Future Work

6.2.1 Possible Improvements

In the future, several improvements can be made to the AIDA system to make it more effective and versatile. Some of the possible expansions and modifications could be done at the hardware level, while others would mostly involve additional software development.

Currently, the manipulation of the motors requires the cooperation of several components; the Android application must broadcast a message that must be received by an R1D1 program in the phone, which communicates with an external computer, responsible for sending the pertinent commands to the circuit boards via USB. This intricate process could be simplified if the phone had direct control over the motor controller boards installed in the robot's body.

At this time, it is rather difficult to achieve a Bi-directional serial communication using the Android phone's port. One directional communication is feasible, but in this project, the phone must send commands to the motor controller boards and it needs to receive the feedback sent back by the encoders. Nevertheless, the phone can use its Bluetooth connection to exchange data with the boards. This alternative is being explored by other projects at the Personal Robots Group through customized motor controller circuit boards. Modifying the AIDA system to employ these new boards would make the robot and the phone completely independent of the external computer, and it could result in a significantly less complicated system.

Another potential addition to the system will be a mood recognition system that will allow AIDA to learn from previous user feedback and have more appropriate responses to the driver's actions. In our opinion, a system capable of recognizing and learning from the user's moods will result in a more personalized agent as the amount of driver-robot interaction increases. A short-term advantage would be the possibility of tailoring AIDA's responses to be more in tune with the driver's moods, possibly

improving driving performance. On the other hand, a stronger bond could be fostered and used to explore long-term user-agent relationships. Having continuous feedback from the driver would allow the system to mold itself as the user's behaviors and preferences change with the passage of time.

Implementing new behaviors that could be activated on a wider range of circumstances, and developing responses to more user commands, would potentially enhance the users experiences and their acceptance of the system. Given the flexibility of the framework, this is likely to be the less involved extension, as it does not require any extreme changes.

6.2.2 Evaluation

Even if the system is not improved or expanded, the natural next step of this work would be to perform a user study to measure how efficient and engaging is the agent that has been developed so far. In a potential study, a group of participants could be asked to perform a series of tasks related to the driving environment; they would follow a predefined script that would include all of the conditions that trigger the behaviors implemented in AIDA. Another group could be asked to perform the same tasks without AIDA's assistance.

The participants' behavior can be analyzed to determine if the agent's assistance induces safer driving practices. For instance, a good measure of distraction would be how often and for how long drivers take their eyes off the road. Moreover, a questionnaire can be designed to assess user satisfaction with the new interface. The examination could inquire about the system's effectiveness and usability, and it could ask the users to provide suggestions to enhance the agent's behavior. These results could be used to revise and make improvements on the current design, if another version were to be developed. Moreover, if the feedback is favorable, it could encourage developers to build innovative interfaces inspired on friendly robotic assistants.

Appendix A

Head Modification Calculations

In this section, we include the calculations made to design the new head shell.

Gearhead: 23/1
Stall Torque: 46.8mNm
Recommended Torque: 10mNm

Using these values, together with the gearhead, we calculate the following torques:

$$\text{Maximum Torque} = \frac{23}{1} * 46.8mNm = 1076.4mNm$$

$$\text{Recommended Torque} = \frac{23}{1} * 10mNm = 230mNm$$

Then, we calculated the maximum weight that could be supported and lifted by the neck to make sure that the current components would work properly after attaching an Android phone. The neck length is 17 cm, and the distance between the joint connected to the head and the one fastened to the base is 12 cm. Assuming that the neck was weightless, then the maximum weight that the head could have if we wanted to stay within the recommended limits is:

$$\text{Recommended Weight} = \frac{230mNm}{120mm * 9.8m/s2} = 0.20kg = 200grams$$

Given that the Epic 4G phone weights around 156g, then the sum of all the other head components should be less than 44g. Even though this is somewhat unlikely, especially because it was assumed that the neck was weightless, the system would still work because these numbers are well under the limits set by the stall torque. Also, these numbers correspond to the maximum exerted torque, and are only valid when the robot is in the seated position, with its neck parallel to the ground.

For the head design calculations, we decided to use about half of the maximum recommended torque, in an attempt to achieve smoother movements. The resulting head length was then:

$$\text{Recommended Head Length} = \frac{230mNm}{2 * .156kg * 9.8m/s2} = 75mm = 7.5cm$$

Bibliography

[1] The bureau of transportation statistics, July 2011.

[2] Communicar - communication multimedia unit inside car, July 2011.

[3] Aide - adaptive integrated driver-vehicle interface, fraunhofer iao, July 2011.

[4] Idc worldwide quaterly mobile phone tracker, June 2011.

[5] droidthing, July 2011.

[6] Google product search - universal android standholders, August 2011.

[7] Dc-micromotors. series 2232 ... sr, October 2010.

[8] L. Andreone, A. Amditis, E. Deregibus, S. Damiani, D. Morreale, and F. Bellotti. Beyond context-awareness: Driver-vehicle–environment adaptivity. from the comunicar project to the aide concept. In *Proc. 16th IFAC World Congress*, pages 4–8, 2005.

[9] R. Bose, J. Brakensiek, and K.Y. Park. Terminal mode: transforming mobile devices into automotive application platforms. In *Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 148–155. ACM, 2010.

[10] Y. Cao, F. Van Der Sluis, M. Theune, A. Nijholt, et al. Evaluating informative auditory and tactile cues for in-vehicle information systems. In *Proceedings of*

the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications, pages 102–109. ACM, 2010.

[11] A.W. Gellatly, C. Hansen, M. Highstrom, and J.P. Weiss. Journey: General motors' move to incorporate contextual design into its next generation of automotive hmi designs. In *Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 156–161. ACM, 2010.

[12] C. Breazeal M. Siegel. Audi sociable car project report year one, September 2009.

[13] Darren Murph. iphone 4s hands-on! - engadget, December 2011.

[14] C. Nass and S. Brave. *Wired for speech: How voice activates and advances the human-computer relationship.* MIT press, 2005.

[15] C. Peter. *Affect and emotion in human-computer interaction: From theory to applications*, volume 4868. Springer-Verlag New York Inc, 2008.

[16] C.D. Wickens. Multiple resources and performance prediction. *Theoretical issues in ergonomics science*, 3(2):159–177, 2002.