DAISY : Dynamic Assistive Information System

Project Report

Richard William Kirk
(rwk01@doc.ic.ac.uk)

Supervisor: Dr Sebastian Uchitel
Second Marker: Dr Alessandra Russo

Wednesday 15th June 2005
Abstract

Sometimes we take for granted how easily we can get around. Many people do not find it so easy and have great difficulty in finding their way from one point to another. Daisy will solve this problem giving such people reassurance and assistance in order to help them to successfully navigate the public realm.

We look at what is required to allow people with learning difficulties to navigate the public realm. By considering the needs of the users and consulting with the Accessibility Research Group from UCL we aim to provide a truly accessible solution.

An application has been provided to display media rich instructions to best describe routes between points. By presenting routes using clear interfaces pre-trip planning is possible. Dynamically using co-ordinates from a GPS device to track the positions of the system along routes allow in-trip reassurance. Support tools have been provided to specify clear routes and build up stores of routes to give users choices about where they go, maximising their independence.
Acknowledgements

Although I would like to take all the credit/blame for this project I feel I have to lay a little responsibility on to a few other people before we go on.

First of all my supervisor Dr Sebastian Utichel, thanks for all the support and guidance through the project at a time when you had a much more important person to be taking care of. Good luck with fatherhood!

For her reassurance at a time when I wasn’t sure if Daisy was actually going anywhere thank you to my second marker, Dr Alessandra Russo.

To the members of the Accessibility Research Group at UCL, thank you for making trips over to Imperial to meet with me and for giving me such an interesting problem to tackle. I have to mention by name Marcels Wainstein for giving such a good specification for the project and Nick Tyler for coming up with a lot of good ideas. A big thank you for the loan of your equipment, especially after I broke my own phone the test phone provided became extremely useful (it was better than mine anyway).

Another group of people who need a lot of thanks are the many people who have developed using J2ME before me. I can’t mention you all by name, mainly because I don’t know who you are, but your posts in various forums have been invaluable and I would never have worked out anything without the records of your experience that you have left behind you. Hopefully lessons I have learnt will help developers in the future.

Thank you to Chris White from MacroSpace for taking time to answer a few of my questions on mobile games development.

For her help testing and then giving feedback on Daisy everybody give a big hand for Shiv!

For help with report writing don’t thank my high school English teachers but do thank Adam for introducing me to \LaTeX. All I can say is at least it looks good!

On a more personal note thank you to all my friends and family for not interrupting me (much) when I worked. Apologies to Vicki for forgetting to call when I stay up all night working. Finally, thanks to my Dad who taught me everything I know...apparently.
Chapter 1

Introduction

DAISY (Dynamic Assistive Information System) is a project which aims to help people with learning difficulties navigate around the public realm and thus achieve a greater level of independence. A mobile application (for use on devices such as a mobile phone or PDA) is required that will use precise location information to feedback to the user their current position. The system will be able to give the user appropriate directions based on a predefined route.

Figure 1.1: Daisy Logo

1.1 DAISY Motivation

The system has been requested by members of the Accessibility Research Group at UCL. They work towards finding ways of making the public realm more accessible for people with all types of disabilities. Recent research that they have been involved with includes:

- Specification for accessible bus stops.
- Study of how pedestrians and buses interact at bus stops.
- Provision of public transport in rural areas.
- Developing better training at university level and a disability and transport awareness in schools.

1 The “Y” in the DAISY acronym does not stand for anything. I guess you could say that it stands for “whY is this Y here?”

2 http://www.cts.ucl.ac.uk/arg/
All of the projects that they are involved in attempt to increase the independence of people with cognitive difficulties. By providing them with assistive tools people with disabilities are given the support and confidence they need to go out and face new challenges.

The DAISY project is funded by the Department for Transport. The specification of the project can be seen on-line on the Department for Transport web-site.

1.1.1 DAISY Objectives
The objectives for the project, issued by the Department for Transport, are:

1. Enhance independence of people requiring pre-and in-trip information about pedestrian and public transport journeys.

2. Confirm benefits to people with learning difficulties within the frame of a larger sample and a greater number of contexts than was possible in the previous feasibility project (DIMPLE).

3. Conclude pre-development research of the video map investigated in DIMPLE.

4. Research information about pedestrian environments and how to incorporate this into the video map.

5. Develop multi-terrain application (e.g. building sites) where visual cues can help.

6. "Industrialise" the DIMPLE software to make it easier for users and travel trainers to develop their own journeys using the software, both in-trip and pre-trip.

The DAISY objectives tackled by this project will be to provide software to provide visual clues and also allow users and travel trainers to develop their own journeys. Objective 1 must be satisfied by the software application. The majority of the work on confirming benefits to the people with learning difficulties and relating this to work done on the DIMPLE project (see section 2.2.1) will be handled by the members of the Accessibility Research Group at UCL.

1.2 Project Content

1.2.1 Contributions to DAISY
This project contributes software to allow the dynamic visualisation of routes. Routes can be chosen depending on the users start point and desired destination. The routes can then be viewed either by manually searching through a route or by using a tracking feature that automatically moves the user through a route using precise location information from the global positioning system (GPS, http://www.rmd.dft.gov.uk/project.asp?intProjectID=11119

Details of the DIMPLE project are given in section 2.2.1.
see section 2.3.2. Manually viewing routes allows pre-trip planning where as automatically tracking the user through a route gives them in-trip reassurance.

In addition to being able to view routes, an administrator system has been provided that allows routes to be specified. A travel trainer can go out and collect route details, including media to help describe the route, which can then be followed by a user. Daisy is capable of capturing audio and image media which is then used to describe the directions in a route.

Using the administrator system and a docking station on a desktop PC it is possible to build a large store of routes. Routes and media files associated with them can be transferred to and from mobile devices using Bluetooth. This overcomes memory restrictions on the mobile device providing the user with numerous routes to different destinations such that they have increased freedom.

1.2.2 Main challenges faced

The most technically challenging parts of this project have been:

**Connecting to a GPS to use tracking:** first of all a connection had to be established between a mobile phone and a GPS device using Bluetooth. Once the connection was created NMEA-0183 location data had to be taken from the stream to find the latitude and longitude of the systems location. This then had to be integrated into a tracking system to which other parts of the system could connect and register to receive location updates. More details on this feature can be seen in section 7.5.

**Media:** in order to best describe routes different types of media are required. This media must be accessible on the mobile device where memory is limited. Getting media data onto the mobile device is difficult as there is no access to the local file system and only slow, intermittent network access. An alternative to transferring data onto the device is to use the device itself to capture media of different types. Once the different types of media are loaded into routes to be followed by users, the media then has to be displayed appropriately. Media implementation is included in section 7.3.

**Creating a docking station to transfer Daisy data over Bluetooth:** this became particularly challenging due to restrictions on the Bluetooth implementations. The docking station allows data to be sent between desktop machines and mobile phones fitted with the appropriate Bluetooth hardware and docking station software. The docking station service can be offered on a number of devices which are found using local device discovery. More information is included in section 11.5.
1.3 Report Content

This report is laid out in four main parts.

**Background** to the project giving the history of the project and the requirements set out by the accessibility research group. This is the foundation stone of the project and represents the work done prior to development of the Daisy application. (Chapters 2 & 3)

**Technical report** including details of the development of the carried out for the project. As well as an overview of the Daisy application this will include technical details of the approaches used and justification of design decisions. During development feedback was given by members of the research group and this feedback was used in later development, the feedback given and how this affected progress has been included. (Chapters 5 to 11)

**Evaluation** of the final system which will include details of both the successes and failures of the system. This will be a chance to look back and say what has worked in the system and what could be improved upon. User testing has been included in the evaluation to better understand the operation of the system. (Chapter 12)

**Conclusions and Future Work** will be presented in the final section which will summarise the work done and suggest ways in which it can be continued in the future. (Chapter 13)
Part I

Background and Specification
Chapter 2

Background

Before a solution for Daisy could be developed it was important to fully understand the history of the problem, why it is needed and the desired results. This chapter gives an overview of some issues that influence Daisy.

2.1 DAISY: Dynamic Assistive Information System

Before creating a Daisy system it is important to fully understand the motivation behind the system, in particular who will be using it and why they need it. During meetings with members of the Accessibility Research Group from UCL the background to the problem was established.

2.1.1 Potential Users of DAISY

DAISY intends to increase the capabilities of those with learning difficulties. The system aims to break down barriers faced by its users allowing them to navigate the public realm successfully. It should give the users a greater sense of independence reducing the amount they need to rely on others, allowing them to lead full lives and taking part in whatever activities they choose to.

There are a number of different aspects to the cognitive difficulties that create barriers for the users. These include:

- **Memory disabilities** where the user often does not remember what has previously been seen or heard. It is difficult for people with such disabilities to remember sequences of directions or instructions. This may result in the same error being repeated frequently.

- **Perception disabilities** are shown when someone has difficulty distinguishing between different positions and shapes, judging distances and depths and also may have a poor sense of direction. In addition there problems with audio perception can be experienced where distinguishing different sounds is difficult. Overall this area concerns the inability to identify and integrate information.
Problem solving disabilities occur in some individuals where once faced with a problem they may be reluctant to deal with it. Very often this can lead to extreme frustration and generally entails giving up on rather than persisting with the problem.

Conceptualising disabilities can include problems in sequencing and categorising information, applying previously learnt information, thinking about cause and effect, and considering abstract concepts. Often such problems become apparent through difficulties in skill development.

These different cognitive difficulties are often results of conditions including autism, attention disorders, dyslexia, cerebral palsy, epilepsy and traumatic brain injury. Cognitive disabilities are the most common form of disability with more than 4 times as many people experiencing them to some degree than those that suffer from blindness.

Cognitive disabilities must be considered when forming a solution to the problem at hand. The DAISY system must provide a clear navigation system that its users are capable of understanding and following. Consideration must be taken for the form of input and output used in addition to giving clear, understandable and easy to follow instructions.

2.1.2 Scenarios faced by DAISY users

A wide range of difficult situations are faced by DAISY users that this system aims to help them deal with. Some example scenarios are given here in order to provide some insight into precisely why DAISY is needed.

Getting lost during familiar routes. Even if a user follows a route regularly the cognitive difficulties that they suffer from may cause them to forget which way they should go during the route. Their memory disabilities may mean that the user needs frequent reminders of where they should be going in order to reach their destination.

Being close to destination but not close enough. Often users are close to where they need to be however not close enough. It is possible that they are the wrong side of a building or on a street adjacent to where they need to be. It is needed that in these situations the users can be given a direction to where they actually need to be.

Dealing with unfamiliar routes. The users often struggle with problem solving and hence finding their way to a destination along an unfamiliar route can pose real problems. To reach their goal it is necessary that they be given frequent directions.

Using public transport and travelling as a pedestrian. When travelling on public transport users need to be given instructions on where to get on and off the bus. They should also be given information as which bus to get, this information may be presented in the form as a photo of a bus stop where they should get the bus or a photo of the bus that they should get on.
Users panicking when they lose their way. If the users become disorientated they often panic making the situation worse. In this situation it is desired that they can request help. Functionality is required so that they can easily contact a helper via the DAISY application allowing them to quickly gain assistance and once again find their way.

2.1.3 Providing a Solution for the Users

Assistive Technology

Assistive Technology refers to devices and equipment that are used to maintain, increase or improve the capabilities of people with disabilities. Such equipment can be used to increase capabilities that have been limited by a range of disabilities including mobility, hearing and visual disabilities and learning difficulties.

Examples of assistive technologies include:

- Wheelchairs
- Canes
- Hearing aids
- Computer Access Solutions
  - Voice input and output
  - Increased font and image sizes
  - Adapted input devices

These Assistive Technology Products help people with disabilities to accomplish daily tasks, communicate with others, have the same chance as others in gaining an education or being able to work, and allowing them to participate in recreational activities. In general they aim to help people achieve a greater level of independence and enhance their quality of life. DAISY will provide a similar function by assisting the users in navigating the public realm.

Trip Planning

Two important aspects that need to be considered are pre-trip planning and in-trip reassurance. Pre-trip planning allows the users to have a carefully set out a route that they will follow. The routes will be described by an administrator who will load the route onto each users mobile device. The user will be able to preview the entire route before they set out in order to familiarise themselves with where they should be going.

The route will be available during the trip in order to let the user that they are on track. The DAISY system will provide this in-trip reassurance by telling the user exactly which point of the route they are on, let them know whether or not they are in the right place, and also to show them where to go next.
2.2 Existing Work

2.2.1 Previous Projects

Dynamic Information to Mobilise People with Learning Difficulties (DIMPLE)

DIMPLE aimed to investigate the feasibility of using video clips as a means of providing journey information to people with learning difficulties. An investigation was carried out with a group of people with learning difficulties. This determined the best forms of visual stimulus and ways in which information and instructions can be presented in order to make them more independent when making journeys.

The work from this project will be extended into the DAISY system. The different visual stimulus decided to be most effective included photographs, cartoons and videos. DAISY will provide a means of presenting this information when required.

The project was developed in conjunction with Hackney Community Transport \(^1\) and funded by the Engineering and Physical Sciences Research Council \(^2\).

Previous work on DAISY

Attempts have already been made to produce the software the will be produced in this project. These past attempts at the DAISY project were unsuccessful due to the methods used to obtain location information. The methods used to obtain location information included:

Radio beacons which were found to be very difficult to connect to and did not give accurate enough location information.

GPS was been used in an attempt to solve this problem however technology available at the time was either very costly or unsatisfactory to solve the problem due to poor resolution on location finding and poor satellite signal reception. With recent advances in available GPS devices providing improved signal reception in portable form a new attempt will be made at solving the problem using this method.

A further problem with creating a suitable DAISY system was the device on which it would be developed. Until recently portable devices have been expensive, had low processing capability and were limited by their displays. Mobile phone and PDA development now provide relatively high powered devices capable of running a wide range of applications. New screens are available replacing small monochrome displays with high resolution colour displays capable of showing high quality photographs and videos. These advances make possible the development of a successful DAISY system that will be accepted by users.

---

\(^1\)http://www.transportforall.com/hackneycommunity.asp
\(^2\)http://www.epsrc.ac.uk/
2.2.2 Mobile Navigator Software

There are a number of pieces of navigator software currently available in the marketplace.

**Wayfinder** is a piece of software offered together with subscriptions to a mapping and direction service. The software runs on a portable device, typically either a mobile phone or PDA, which needs to be connected to a device capable of retrieving GPS information. The software communicates with a server maintained and updated by Wayfinder. Users send requests in the form of an origin and destination address which are used to compile a route between two points. The directions are returned to the software and presented to the user in step-by-step direction form and illustrated on a map. If a user strays from the specified route a request for a new route is issued to the server providing an updated route to the destination.

This application is not suitable as it stands for DAISY as the information is not presented in a suitable format. The mapping and directions can be confusing to the intended users of DAISY as they contain a lot of complex information in an unsuitable format.

More information on this system can be found on-line on the [Wayfinder](http://www.wayfinder.com/) website.

**Power Navigator** is produced by SymbianWare and offers similar functionality to the Wayfinder software. The difference is that this software does not offer a subscription to a mapping service. Maps are purchased and then stored locally on the device. An alternative is to create your own maps using an associated piece of freeware software. This allows you to specify your own routes between locations and add point of interest to navigate to.

Again this software is not suitable as it does not provide the directional information in a suitable format. It is required that the directions are presented using multimedia formats rather than just showing a direction on a map.

More information on this system can be found on-line on the [SymbianWare](http://www.symbianware.com/) website.

**Citymaps** provides detailed mapping of cities around the world. These maps are purchased and loaded into the software for example I have used the system with a map of London. As with the other applications discussed here it is possible to draw directions onto the map from a source point to a destination. GPS information can be used to track progress along the route.

The same problems for users of reading maps are presented by this system as previously mentioned making this an unsuitable solution.

---

3http://www.wayfinder.com/
4http://www.symbianware.com/
Dedicated GPS Devices

In addition to this software designed to run on a number of devices there are also GPS solutions provided on dedicated devices. These devices offer a complete GPS solution as they are a receiver as well as a device to display tracking information. The devices are aimed at a wide range of users to assist in navigation for maritime, automotive and outdoor pursuits.

Maps are loaded onto the devices from a PC. These maps are then used to track your current position. As with the software above you can specify start points and destinations as well as points of interest along your route. Routing algorithms in the software use the maps to determine step by step directions for the user to follow.

These dedicated devices are not suitable for DAISY as they do not allow the required level of customisation. It would not be possible to replace the software offered on the device with a tailor made application for DAISY. This would mean that the mapping provided with the device would have to be used rather than the multimedia approach to route tracing as is required in this project.

2.3 Background to Technologies Required

2.3.1 Human Computer Interaction

A major part of developing a successful solution for the DAISY project will be ensuring that a suitable Human-Computer Interaction methods are employed.

A lot of work is currently being put into HCI research. In particular a large field of growth is ensuring that computer systems are accessible to all. Groups such as the Web Accessibility Initiative\(^5\) are studying how to make the web accessible to everyone. A lot of the work done in this area can be linked to how best to make software accessible in general.

Accessible Software

The software designed for DAISY must be accessible to people with cognitive difficulties. To cater for people with these needs design principles such as use of consistent layout and simplified language must be used. Often redundant information can be useful such as giving an audio and video file to communicate the same point. This redundancy makes use of a number of skills, in this case visual and auditory, increasing the chance of the user remembering the information presented.

There are a number of common ideas that are generally adhered to when producing accessible software. Here I will discuss the ideas given on the seven precepts of usability and accessibility proposed by TechDis (Technology for Disabilities Information Services)\(^5\). Although these ideas were written with accessible web pages in mind they can be easily transferred to considering accessible software.

Navigation and layout: First of all, the navigational structure must be de-
pendable. This means that it must be possible to move through all sections of the software without failure or errors arising during the navigation. There should be clear hints to help the user to follow different paths through the software such as obvious symbols or clear text.

The layout of each section of the software should be based on common design which should be consistent throughout the system. This consistency makes the system more familiar to users and reduces the requirements for remembering how to use each component of the system. If all information is presented in the same way it is more likely they will remember where to look for what they require rather than repeatedly having to study different layouts.

Such consistency will improve the users recognition of the system and reduce chances of confusion. By reducing confusion it is more likely that the user will tolerate the system rather than quickly becoming irritated by it and losing interest in using it.

Visual presentation and customisation: Ideally user interfaces should be clear and intuitive in order to make the user feel comfortable when using them. This may involve minimizing complexities in the design such as reducing the number of colours or fonts used. The interfaces do not have to be dull, however they should not include excessive details that would distract the user from the actual purpose of the system. Textual information density should not be greater than 50% of the screen to prevent overloading the user with information.

Ideally the user should be able to alter the display settings from within the system. This may be as simple as increasing font sizes but may also involve reducing the amount of colour in the system. Ideally this feature should be added in some degree in order to allow the users to adjust the system making it comfortable for them to use.

In order to make text as readable as possible appropriate typefaces and size of lettering should be used. Flickering or moving text should not be used as it makes it far more difficult to read. Text and the background on which it sits should be contrasting colours making the text stand out. Colour blindness issues, such as not using red and green together, should be kept in mind when selecting colour schemes.

Text descriptions for images: This precept is more aimed at systems which will be displayed on a number of browsers some capable of displaying images and other not. For this system it is essential that pictures and other media be displayed and a suitable environment will be used to enable this. Despite this a text description of images should still be given as it gives a source of redundant information. As previously mentioned this redundant information increases the chance of the user remembering what was shown.

Accessible elements: The point of this precept is that all elements should be checked to ensure they contain all required captions, labels and alternative text if they are not displayed. A similar idea must be applied must be applied in accessible software design in order to ensure that there are no problems displaying form elements on user interfaces and all required the information is presented to users.
In the case of websites this can be checked using a variety of HTML verification tools however in the case of accessible software the checking must be done manually by the developer.

**Use and presentation of written language:** Written language should be presented in good, clear English. It should be kept simple and understandable using a clear structured format. Highlighting should be used liberally and text should be fully and correctly punctuated. Paragraphs should be kept concise and should be associated using distinctive headings. Lists are useful to reduce the amount of information presented and ensure concise presentation of information.

**Accessible issues for other media types:** Where media is used (e.g. audio or video) it is often useful to provide an alternative text representation such as a caption. It is important to ensure that media is stable and will always be available. Missing media will cause a lot of confusion in DAISY. It is recommended that the use of new media formats should be avoided unless it is certain that the format is supported and will be successfully presented.

**Help, searches, errors and documentation:** Help functions are required to assist the user if they struggle to perform functions in the system. Help should be provided for all functionality. It may be useful to provide illustrations on how to perform tasks rather than simply describing processes in words to make the information clearer and more memorable for users.

If errors occur in the system it will not always be useful to present them to the user. In general the users will not be able to cope with the error messages and they will only serve to confuse. To prevent this confusion errors should be dealt with internally by the system perhaps by retrying an operation or performing alternative actions. Ensuring an error free environment will be a part of ensuring that the system is robust.

The system should come with a user guide. This user guide should describe how all aspects of the system operate and how the user should activate them. The user guide should be closely associated with the help details provided by the system.

### 2.3.2 GPS Systems

The Global Positioning System was originally developed by the US Military but has now been extended to civilian use. GPS provides coded satellite signals which are picked up by a GPS receiver. This enables the receiver to calculate their current position, velocity and time. Signals from four satellites are required to calculate a position in 3 dimensional space and the GPS time. More satellite signals are desirable to perform error checking to ensure an accurate value is given. There are 24 satellites orbiting the earth every 12 hours. The orbital tracks provide a user with visibility of between 5 and 8 satellites from any point on earth.[6]

As GPS is developed and maintained by the US military they also have certain methods of controlling its use. Originally Standard Positioning Service (SPS) was only available to civil users giving 100m horizontal accuracy, 156m
vertical accuracy and 340 nanosecond time accuracy due to Selective Availability imposed by the US military scrambling signals. This was removed in May 2000 allowing civilian users access to the 20m to 30m accuracy that was previously only available to the US military. The removal of the scrambling that created SPS was done for a number of reasons including improved use during rescue operations and also for aviation safety. Differential GPS has always been used to improve accuracy using software and hardware methods. GPS systems now offer civilian users the potential accuracy of down to 1m.

The data from a GPS device can be used to calculate the user’s current position and by frequently recalculating position can also give the velocity and direction of travel. This has been used in mobile navigators to find both a user’s location on a map and also which way they are travelling on a road in order to give accurate directions.

GPS Connections

DAISY will be required to connect to a variety of different GPS devices using the NMEA (National Marine Electronics Association) 0183 Output Protocol. The standard defines signal requirements, transmission protocol and specific sequence formats. The output protocol transfers data across a 4800 baud data serial bus. The settings for the protocol are included in table 2.1.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baudrate</td>
<td>4800</td>
</tr>
<tr>
<td>Data bits</td>
<td>8 (Bit 7 set to 0)</td>
</tr>
<tr>
<td>Stop bits</td>
<td>1 or 2</td>
</tr>
<tr>
<td>Parity</td>
<td>none</td>
</tr>
<tr>
<td>Handshake</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 2.1: Serial Settings Table

The NMEA messages are formed into sentences which are transmitted from a 'talker' to a 'listener' which can be a maximum of 80 characters long. Sentences must always begin with a '$' and end with a carriage return then line feed character. The first five characters of the sentence are the address field indicating what data is contained in the comma delimited data that follows. This format is used to allow simple parsing of the sentences.

The 5 character record address is further divided into a 2 character talker id and 3 character sentence type. In this case we are concerned with receiving sentences from a GPS so the talker id will always be 'GP'. The sentence used to communicate GPS data are given in table 2.2.

Table 2.3 shows some sample NMEA messages.

These messages are provide to give an idea about the make up of the NMEA messages so a full description of the syntax will not be given. By studying the messages you can however fairly easily extract the position which was 18 degrees 24.6437 minutes north and 077 degrees 06.8707 minutes west (on the Jamaican coast). Details of the syntax of each message can be seen in the Communication Technology specification of NMEA output messages[7].
<table>
<thead>
<tr>
<th>NMEA Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGA</td>
<td>GPS fixed data</td>
</tr>
<tr>
<td>GLL</td>
<td>Geographic position - latitude/longitude</td>
</tr>
<tr>
<td>GSA</td>
<td>Active satellite information</td>
</tr>
<tr>
<td>GSV</td>
<td>Satellites in view</td>
</tr>
<tr>
<td>RMC</td>
<td>Recommended minimum GPS information</td>
</tr>
<tr>
<td>VTG</td>
<td>Course over ground and speed</td>
</tr>
</tbody>
</table>

Table 2.2: NMEA-0183 Output Messages [7]

<table>
<thead>
<tr>
<th>NMEA Record</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGA</td>
<td>$GPGGA,223355.504,1824.6437,N,07706.8707,W,1,07,1.3,-66.9,M,,,0000*0A</td>
</tr>
<tr>
<td>GLL</td>
<td>$GPGLL,1824.6437,N,07706.8707,W,223355.504,A,A*4D</td>
</tr>
<tr>
<td>GSA</td>
<td>$GPGSA,A,3,10,07,24,04,17,05,13,,,,3.1,1.3,2.8*3D</td>
</tr>
<tr>
<td>GSV</td>
<td>$GPGSV,3,09,28,06,158,*40</td>
</tr>
<tr>
<td>RMC</td>
<td>$GPRMC,223355.504,A,1824.6437,N,07706.8707,W,0.00,,291204,,,A*68</td>
</tr>
<tr>
<td>VTG</td>
<td>$GPVTG,,T,,M,0.00,N,0.0,K,A*13</td>
</tr>
</tbody>
</table>

Table 2.3: Sample NMEA-0183 Output Messages collected on the coast of Jamaica

2.3.3 Bluetooth

Bluetooth is the industry standard for low power, short distance radio links. It enables wireless data connectivity between different types of devices such as laptops, mobile phones and digital cameras. Low power Bluetooth has a range of around 10m whereas higher power devices can be operated at ranges up to 100m. The devices operate in the 2.4GHz license free Industrial, Scientific, Medical (ISM) frequency.

The initial use for Bluetooth has been to replace cables when connecting peripherals to a computer system. Protocols have been developed for use with Bluetooth including business cards, file transfer and dial-up networking. The advantage of Bluetooth is that it works silently and automatically in the background.

“Bluetooth” was the nickname of Harald Bløtland II, king of Denmark from 940 to 981, who united all of Denmark and part of Norway under his rule. A runic stone has been erected in his capital city, Jelling. The runes say:

“Harald Christianized the Danes
Harald controlled Denmark and Norway
Harald thinks notebooks and cellular phones should communicate seamlessly” [5]

Bluetooth is mentioned here as it has been recommended that DAISY should use the standard to connect to a GPS system from the mobile device that it runs on. This will allow a quick and easy method of communicating between the devices removing the need for the users to perform complex set up procedures.

A serial connection will be established using Bluetooth.
tion will be used to receive NMEA output messages from the GPS device. The DAISY system will need to connect to the Bluetooth serial port service that is offered by the GPS device before it can receive any messages.

### 2.3.4 Mobile Development Platforms

The desired mobile development platform to be used is the Symbian Series 60 Operating System. Even though the use of this platform is required at this stage it is important that the system be designed in such a way that it can be adapted to other platforms in the future whether they be newer versions of the operating system or platforms on different kinds of devices.

The Series 60 Developer platform is built on Symbian OS Technology. Symbian OS is designed specifically for mobile devices which are typically small, power and memory constrained devices requiring use of a range of communication technologies. By providing a wide range of features, connectivity and development environments, the Series 60 platform has been very popular and is used on a number of devices from a number of manufacturers all based on a common set of standards.[9]

#### User Interfaces

Developers are free to specify their own user interfaces however Symbian OS does specify a style that should be adopted. There are also libraries available which promote a common look and feel in all applications on the device. This is suitable for DAISY as it will provide a common layout that will enhance usability by removing the potential for confusion to users.

### 2.3.5 Available Development Languages

In terms of application development Symbian OS Series 60 provides C++ as the native development language as well as Java APIs. It is possible to migrate applications developed for the Series 60 to other developer platforms such as Series 40 and 90 as well as to platforms on other devices based on Symbian OS. All of these are capable of running both the native Symbian OS C++ and Java MIDP. Provided applications are developed without making use of device specific software migration from one platform to another should be relatively problem free with only minor issues expected in user interfaces and input.

#### C++

The native C++ on Symbian OS has been optimised to cope with the memory constraints on the devices that applications are expected to run. Public APIs are available to harness all of the features that the device provides including Bluetooth, multimedia, graphics and user interfaces.

As the underlying operating system was written in C++ the language is a strong development choice for third party developers. The language is capable of using all features offered by the device including wireless communication protocols including SMS and Bluetooth and use of the phones built in features such as the camera[10].

16
As the C++ used for Symbian OS has some variations to other versions of C++ applications developed may be very difficult to port to non-Symbian devices. This could pose major problems should the application be required to run on devices on the market using platforms such as Windows CE rather than Symbian OS. Such C++ programs written in the native Symbian C++ may need completely rewriting before they will compile for use on other platforms.

Java

Recent figures from Sun Microsystems predict that Java will be available on over 600 million mobile phones shipped worldwide in 2005. This makes Java a strong contender as a development language due to its cross platform portability that should make the application easy to move to other devices.

The version of Java used on mobile phones is called J2ME (Java 2 Platform, Micro Edition). This version of Java was designed specifically for consumer and electronic devices which have constrained resources (both memory and power). J2ME offers a number of configurations for different hardware devices. As DAISY is designed primarily for a mobile phone it will use the Connected, Limited Device Configuration (CLDC). This is designed for battery powered devices with 160-512 KB memory available for Java that are often slow and have an intermittent network connection. On top of the CLDC is the Mobile Information Device Profile (MIDP) which specifies an API set appropriate for mobile phones. MIDP version 2 allows the use of Bluetooth as required in DAISY.

Despite the advantages in portability Java does have drawbacks compared with native applications in performance and functionality. The overhead of using a Java runtime environment to execute applications decreases performance and using the standard Java APIs means that some specialised features offered by hardware cannot be fully utilised. Despite this Java is still suitable for DAISY as all required functionality is provided and performance of a Java application should be satisfactory for the users as speed is not the main factor in the system.

Development Tools

SDKs are available for both C++ and Java development provided by Symbian, Nokia and Sun. These SDKs have also been adapted as plug-ins for many popular IDEs such as Eclipse having a Nokia Java MIDP SDK plug-in. As well as the SDKs emulators have been developed which allow testing of all aspects of a mobile application without the requirement of the device itself.

---

*http://java.sun.com/j2me/
Chapter 3

Specification

The accessibility research group had some clear ideas about what they wanted from Daisy. This section formalises these ideas to produce clear requirements for Daisy. All expected results from the project are described here.

3.1 DAISY Specification

DAISY is an information system for people with learning difficulties. The basic requirement is for a piece of software that will help the users navigate around the public realm. The software must retrieve the location of the user from a GPS device. The current location will be used to give the user their next direction towards their destination. The software is to be aimed at users travelling on foot.

The instructions to the users destinations will be given using a series of pieces of media. As they progress along the path new media is displayed from which they will be able to determine the direction that they need to travel in. For example, based on their current position they are given an image of something they must move towards. Once they move closer to this location they are then given a new target.

Media to be presented to users includes photographs, audio messages, cartoon images and videos. Different sets of directions must be loaded into the software depending on where the user intends to go. An administrator will compile the sets of directions and load them into the software.

Administrators will use a different piece of software to acquire routes. The administrator will be required to travel along the routes themselves acquiring the media required in order to compile directions for the users. Following the acquisition of this data it must be compiled into routes that can be used with the first piece of software.
3.2 DAISY Environment

DAISY is required to run on a mobile phone. The phone chosen as a test platform is the Nokia 6600\(^1\) (figure 3.1).

![Nokia 6600](http://www.nokia.com/)

Figure 3.1: Nokia 6600\(^1\)

This mobile phone has been chosen for several reasons:

**Bluetooth:** as the phone has Bluetooth capability it will be possible to connect it a GPS device to obtain location information.

**Large screen:** the Nokia 6600 has a bright 65,536 colour, 176 x 208 pixel TFT display. This will allow the required media to be displayed clearly to the users.

**Camera:** the camera will enable administrators to use the same platform to go out and obtain routing information as is used to display the information.

The mobile phone will communicate with a Fortuna Clip-on GPS (figure 3.2) via a Bluetooth link to obtain precise location information.

![Fortuna Clip-On GPS](http://www.nokia.com/)

Figure 3.2: Fortuna Clip-On GPS\(^1\)

This GPS was chosen as, unlike other GPS units, it supports two modes. One is the standard mode which operates like most other GPS devices and the other is a higher power mode which picks up weaker satellite signals. This feature allows the GPS system to be used nearer large buildings and inside buses where other GPS units would fail to receive satellite signals.

Although the test environment uses these two devices ideally the system developed should be easily moved to other devices including other models and brands of mobile phones and possibly PDAs. Where possible other GPS units should be able to be used with the system to allow future improved or cheaper devices to be used.

\(^1\)http://www.nokia.com/
3.3 User Interface

It is essential that the user interface is suitable for the target users. User interface requirements for those with cognitive difficulties were discussed in the background section (2.3.1). The system must be simple and easy to understand making it accessible to users.

3.4 Milestones in Development

The members of the Accessibility Research Group who tasked this project have given a list of requirements that they wish to be fulfilled. These have been broken down into a number of milestones. Each milestones successful completion results in a deliverable product that will be of use to the users and allow further development to be carried out.

3.4.1 Milestone 1

This milestone is concerned with making the main DAISY user software to operate on the mobile device. A piece of software will be developed to give users pre-trip planning and in-trip reassurance enabling them to successfully navigate the public realm.

GPS connection and control system

In order to obtain location information the application must connect to a GPS device. Menus will have to be developed allowing the users to easily select and connect to a GPS device over Bluetooth. Once the connection has been established the GPS data must be read and interpreted in order to supply an accurate current location. A menu will be needed in the final application that allows a user to see their current location as obtained from the GPS device.

Software for pre-trip planning and in-trip navigation

The pre-trip planning will allow the users to step through a route stored on the mobile device in order to familiarise themselves with it before they set out. This will require list to be presented allowing the users to view each piece of media in turn. They must be able to step through the route forwards and backwards.

When requested the software should be able to retrieve an image based on the current location supplied by the GPS system. This will provide the user with step by step instructions in image form as they move through their journey giving them in-trip reassurance. Each image will have a GPS location associated with it and when the user arrives at this location the image should be displayed.

It is important that this application be robust. The users will not be able to cope well with errors if they occur so must be avoided wherever possible. This will mean that the system must be tested extensively. If the system does fail it is important that the system should be able to be restarted in a state where the users can continue as they left off before the error.
User Customisation of Mobile Application

In addition to the ability to display routes there must be some way of changing the options that specify how the routes are displayed. Settings will be needed to control:

- The required destination of the user
- How long images are displayed for
- What kind of visual information should be displayed (photos, cartoons etc).
- How long each image should be displayed for
- The proximity to a point at which media should be displayed

3.4.2 Milestone 2

Portable route acquisition software

Software should be produced that will allow the administrators to specify new routes and destinations for the users to follow. It must be possible to load these routes into the software created for milestone 1.

In order to gather information that make up the routes to a destination a second application will be developed. This will allow an administrator to follow the route themselves using the mobile phone to gather images and videos. These will be stored as they are taken and when taken a GPS location will be recorded and stored with the image. The routes will have to be stored in a suitable form that they can later be removed from the device and manipulated. A guide should be produced to tell administrators how they can go about this.

3.4.3 Milestone 3

This milestone will build on the work done in milestone 1 on the mobile user software.

Adapt the software to display video information

The existing software should be modified so that in addition to displaying images (photos and cartoons etc.) it will also be able to display videos. These videos will need functions associated with them to 'Play', 'Pause', 'Stop' and 'Rewind'.

Allow multiple pieces of media for the same location

Until this point a given location will only have been allocated one piece of media. It is desired that many pieces of media, for example a photograph and a video, be associated with the same location so that if one piece of media does not help a user find their way the second will. There must be a way of selecting different pieces of media when at a particular location.
Assistance feature

In the event that a user should get lost and not be able to locate themselves based on the instructions given by DAISY there should be an assistance feature provided by the system. This should contact a helper, either using a phone call or text message, so that they can get in contact with the user and offer them assistance.

3.4.4 Milestone 4

Route management for administrators

The mobile application produced in milestone 2 will allow administrators to capture new routes. Once all of this information has been gathered the administrator must be provided with a method of linking these images to form a route to a required destination. In order to link the images into a route a third piece of software that runs on a PC should be provided.

This software will take the data from the mobile application and allow routes to be created from newly acquired data, merged with other routes, modified to insert and remove points on the routes and stored ready to be loaded onto a users mobile device.

The application should provide a suitable graphical user interface from which administrators with little technical background can easily manipulate the routes.

3.4.5 Possible Extensions

Display local mapping and associate this with images displayed

The local mapping will allow the user to see exactly where they are so that they can track their progress and see the distance to their destination. A setting in the software should determine whether or not the mapping is displayed. This will allow users access to more information about their location should they require it and feel comfortable using it.

More powerful routing

Many mobile navigators provide precise, powerful routing between any two points, a source and a destination. An extension for this project would be to provide the same power in routing between such points. This may involve using existing mobile navigator software as a base for DAISY and extracting the routes from this system to form the assistive routes suitable for DAISY users.

DAISY for use on public transport

The initial work on this project will concentrate on users as pedestrians, a possible extension is to adapt the system so that it is suitable to operate on public transport. For example on buses DAISY should track the progress of the bus journey and alert the user when they should disembark the bus. The user will have to be guided to the correct bus stop and instructed which bus they should get on. Similar functionality could be developed for trains and the
underground however these situations add the complication that a GPS signal may not be available.

**Testing on multiple platforms**

The system should be moved onto other platforms including different mobile phones running different operating systems and connecting to other GPS devices. If problems are experienced with running the application on these systems any required modifications should be made with the intention of making a single portable solution even if this comes at the sacrifice of some functionality.

### 3.4.6 Milestones Required

The first three milestones should be completed in full. The fourth milestone has been separated out as it could be a lengthy process to create a suitable graphical user interface to allow full manipulation of routes. As this project is mainly aimed at exploring the potential of the DAISY software it is adequate to provide a guide as to how the routes would manipulated and the interface to carry out the operations can be implemented at a later stage should the project be used further.

There should be consideration of how the extensions should be completed and a report specifying their feasibility should be provided.

### 3.4.7 User Feedback

DAISY has been requested by members of the Accessibility Research Group at UCL. It is important that they are frequently consulted about the progress of the project. As this is a prototype that is being developed they should be consulted at least at the end of each milestone. The feedback that they give should not necessarily be implemented in the project as this may not possible in the given timescale, it should however be included in the final report. Any minor changes requested that will improve the operation of the system should be made provided they don’t disrupt the progress of the project.

### 3.5 Required output from the project

The following practical and written work will need to be produced from the project.

#### 3.5.1 Practical Work

The main practical requirement is an application that will execute on the Nokia 6600. The application must be demonstrated and function as a prototype that can be given to users. Following user tests the specifications will be revised and the system will be adapted to meet any changes needed in requirements. The demonstrated software should illustrate the functionality described in milestone 1. It is important that this application be robust in order to be suitable for the users.
The second piece of software to run on the mobile device will allow administrators to go round and acquire images and GPS data. If time allows a third piece of software to run on a PC will be created to manage the routes.

Any improvements or adaptations made to these systems should be included in a separate application which will be used to show the viability of the improvements. The initial piece of software will be kept as a benchmark to ensure that a robust piece of software is provided without any of the modifications disturbing this. Although not a formal requirement of the project this application may be useful to illustrate the potential for extending the existing system.

3.5.2 Written Work

Extensive documentation will be required to describe DAISY. The system has been requested by members of the Accessibility Research Group and they wish to receive details of all aspects of the system. This report forms a major part of the documentation.

As this will be a new system an in-depth user guide will be required. This will be aimed at the administrators of the systems who will be responsible for instructing the actual users on the operation of the system. The user guide will include a walk-through of the operation of the system introducing each feature and how it can best be used to guide the user through the public realm. As well as the user guide for the user system a user guide will be required for the administrator system and how the routes are composed.

The work on the extensions will need a report detailing the viability of extending the system with the suggested improvements. Other potential developments for the system should also be described in the same report.
Chapter 4

Evaluation Criteria

4.1 Results required from project

Successful completion of this project will require the delivery of the following items:

- DAISY software for users to run on mobile device
- Mobile administrator software to acquire images
- Project report as described in specification
- User guide for both DAISY and administrator software
- Feasibility report discussing the proposed extensions

Also, depending on the time available:

- PC based software to manage routes

The software produced will be evaluated against various criteria, test cases and scenarios in order to conclude whether or not it has met the requirements set out for the project.

4.2 Evaluation Criteria

For successful completion the software produced must provide:

- DAISY user software
  - Accurate location specification and directions
  - Clear, simple interfaces suitable for users with cognitive difficulties
  - Interactive communication with users based on their location and inputs
  - Customisable settings with appropriate controls for presentation of media and directions
- A robust environment that can be relied upon by users
- Portability between mobile devices along with compatibility with multiple GPS systems

**Administrator Mobile Software**
- Mobile software capable of running in the same environment as the user software
- Facilities to acquire media of different types to fully represent directions

**Administrator PC Software**
- Clear software allowing non-technical users to intuitively manipulate routes
- Full capabilities to create, edit and store route data

**User Acceptance**
As the system has been tasked by external users it is important that they are happy with the results of the project. By continually consulting the users on progress of the system and listening to their feedback this should be easily achieved and the system accepted. The final system will be evaluated by the users and their opinions will be included in the final report.

**4.3 Test Cases**
To know whether or not the system functions correctly a number of tests will be used. A report of the tests successes and failures will be given. Should the system fail any tests this will not be seen as an outright failure of the system as long as details of how the system can be improved in order to successfully pass the test are provided.

**4.3.1 Standard Use Tests**
These will ensure that the standard functionality of the system works correctly. Some basic test cases for the DAISY system will ensure the following:

- **DAISY mobile user software**
  - Correct GPS location is established and used in the software
  - A route to a particular destination to be loaded into the system
  - Routes can be stepped through forwards and backwards resulting in the correct sequence of directions
  - At a given location the correct media is displayed
  - Media changes when appropriate as the user moves
  - When off the route no direction is given but a panic/help service is offered
– Multiple items of media at a single location can be viewed in turn
– Media is viewed appropriately depending on format (image, audio, video)

• DAISY mobile image acquisition software
  – Photographs can be captured and stored
  – Video can be captured and stored
  – Locations stored with media are correct
  – Acquired data is stored correctly with location information so that
    is can be formed into routes

• Route management software
  – Routes can be created from data acquired by mobile system
  – Order of nodes in a route can be rearranged
  – Two routes can be merged
  – Multiple routes can be stored and loaded when required

4.3.2 Stress Tests

One of the main requirements of the DAISY application is that it should be robust for the users. To ensure the robustness stress tests will be performed. These tests will run the system in simulated extreme conditions. These will include:

• Quickly moving between locations on the route
• Moving between points causing media to be interrupted, such as moving before videos finish
• Moving on and off the known route
• Turning off the mobile device during the route and then restarting
• Working in areas where no GPS signal is available

As the system must be robust it should be able to cope with all of these situations without failing or presenting any error messages. Further stress tests should be added based on what is discovered during the development of the system.

The administrator software does not need the same stability so will not be given stress tests in the same way as the user software.

4.4 Evaluation Scenarios

The following test scenarios will be carried out by someone not involved in the development of the system. This could either be an independent third party or one of the members of the Accessibility Research Group who requested the system. Either way the outcome of the test will be whether or not the user could complete the scenario and also any feedback that they have after performing the required tasks.
4.4.1 Basic Navigation

In order to test the basic navigation functionality of the system a sample walk-through will be required. This will involve a tester who is unfamiliar with the system being able to start from a starting point and navigate their way to a destination point. In this test scenario the user will be unaware of there destination.

In order to pass this scenario the user must be able to find the correct destination without any external input other than that provided by DAISY. Should the user fail to reach the required destination the system will have failed the test.

4.4.2 Specifying new routes

An administrator should be able to go out and create a new route by capturing images using DAISY. Once this information has been acquired and compiled into a route another user should be able to navigate the route without prior knowledge of the destination as in the previous test. Successful completion of this scenario will allow any route to be added to the DAISY route store.
Part II

Technical Report
Chapter 5

Technical Report Structure

Before starting implementation it was important to look closely at the features available in J2ME, the chosen development language. These features are discussed in the chapter titled “Technical Details” (chapter 6).

Next the “Architectural Issues” (chapter 7) in the Daisy system were considered. This chapter looks into the major design decisions that facilitate the development of the rest of the Daisy system.

The implementation of the Daisy system was split into two parts. The development of the user system and the development of the administrator system. Both use the core set of functionality detailed in the “Architectural Issues” chapter. The administrator system provides a set of utilities that can be used to manage the information required by the user system.

Development of the Daisy user system was split into two stages. Between these two stages a meeting was held with the research group who requested and gave the specification for Daisy. Feedback from the research group was used in the second stage of development to make necessary changes to the user system.

These implementation chapters are divided into:

- “Daisy User System: Stage 1” (chapter 8)
- “User Meeting” (chapter 9)
- “Daisy User System: Stage 2” (chapter 10)
- “Daisy Administrator System” (chapter 11)
Chapter 6

Technical Details

6.1 Programming Mobile Environments

When programming for mobile environments there are a number of restrictions that should influence any design decisions made. These mainly concern the mobile phone available on which you can develop your applications. The smart phones that Daisy is intended for have a number of limitations, example include:

Memory is limited on mobile devices so must be used efficiently. The Nokia 6600 has a heap size of 3MB. Although this seems a lot for a small application it must be considered that other applications will be sharing the heap so the actual amount of memory available to Daisy will be much less. It is more likely that less than 1Mb of memory will be available in the heap for your applications due to the memory footprint of the operating system, Java virtual machine and other applications.

Processor speed is limited on the devices. On the Nokia 6600 Java applications can expect to run at 35MHz. This may seem like sufficient speed for a mobile phones processor however more and more complex applications being developed for mobile devices demand more processing so may still run slowly.

Network connectivity is not always available. The Nokia 6600 is capable of communication using GPRS through a mobile network. This allows transfer speeds of 32-48Kbps. This will allow data to be transferred onto and off the device however it cannot be automatically assumed that this connection will be available. Provision must be made in the event of loss of network connectivity.

Screen size limits the amount of information that can be displayed at once. Most mobile phones have small screens so it important to consider how much information can be shown clearly in the space available. Screen sizes also vary between devices. It is desired that the Daisy application should

\footnote{For more details on what features phones have available a good starting point is Ben-Hui.net (http://www.benhui.net/modules.php?name=Midp2Phones).}
work on multiple devices so user interfaces must be adaptable enough to fit different screens.

**Media capabilities** vary between devices. Some allow audio and video to be played while others are limited to only images. Methods of media capture also vary, some devices have cameras and microphones to allow media capture while others may not.

**User interaction** is usually limited to a few keys on the mobile device. The keys available for use on the mobile phone include:

- Key pad (0 - 9)
- Pound (#)
- Star (*)
- Soft keys (left and right)
- Joystick (directional and fire)
- Cancel button
- Call and disconnect
- Alphabet button
- Menu button

Some more complicated smart phones include additional input capabilities such as full keyboards and touch screens. For Daisy, it is assumed that only the basic set of keys are available for use.

### 6.2 Java Micro Edition (J2ME)

Due to the rapidly growing market in embedded computing, J2ME has been developed to provide Java support for devices such as mobile phones, pagers and personal digital assistants (PDAs). Java virtual machines implemented for mobile devices provide a set of APIs which are a subset of those found in the more commonly used J2SE (Java Standard Edition, for desktop environments) with some additional APIs provided that exploit features specific to mobile devices. J2ME has quickly become the leading platform for mobile development with many applications and games being developed and deployed using the technology. This means that J2ME is already widely available on many mobile devices and is becoming a standard feature for new products.

Java has been chosen as a platform for this project for a number of reasons:

- Portability between different hand-held devices (mobile phones and possibly PDAs in the future)
- Availability of Java implementations on many mobile phones
- Expected future expansion of Java as the standard platform for programming mobile devices
- Ability to exploit device specific functions using a variety of well defined APIs
The main advantage of using Java over the native Symbian C++ is the possibility of portability between phones running different operating systems.

6.2.1 J2ME Architecture

J2ME is built from configurations, profiles and optional packages.

Configurations

The two configurations in use are CLDC (Connected Limited Device Configuration) and CDC (Connected Device Configuration). CLDC is intended for devices with less memory, less powerful processors and intermittent network connections. CLDC is the standard configuration for mobile phones so will be used in this project.

Some packages have been inherited directly from J2SE by the J2ME configurations. Other classes have been specified purely for J2ME. Figure 6.1 shows the how the configurations are a subset of J2SE.

![Figure 6.1: J2SE and J2ME](image)

The core classes in CLDC are implemented in the KVM which is almost entirely compliant with the JVM (java virtual machine). The KVM is designed to run on devices with 160KB to 512KB of memory available for the Java platform.

CLDC Limitations

In order to execute in the limited memory environment CLDC has some limitations compared to other Java versions (J2SE and J2ME CDC). Some of the most important to note are:

- **Floating point numbers** are not supported in CLDC 1.0 (the most widely used version of the configuration). Although support has been introduced in CLDC 1.1 very few devices currently offer this.

- **Limited error handling** is available with only three error classes java.lang.Error, java.lang.OutOfMemory and java.lang.VirtualMachine.
Security restrictions mean some functionality found useful in other versions of java are not available such as the Java Native Interface (JNI), user-defined class loaders and reflection.

Profiles

Where configurations supply the basic functionality for programming any device, profiles are provided to be more specific to a particular type of device. For example, although a pager and mobile phone will have some shared functionality the way they display information to and interact with a user vary greatly. Profiles are provided to complement the configurations and provide distinct capabilities to exploit the diversity between devices. Profiles include:

Mobile Information Device Profile (MIDP) providing core functionality for mobile devices including user interfaces, application life-cycle management and local data storage. It is used on devices such as mobile phones, pagers and PDAs.

Information Module Profile (IMP) for devices with no graphical interface.

Foundation Profile for devices with network connectivity but no user interface.

Personal Profile give full GUI support including the entire Abstract Window Toolkit (AWT).

Personal Basis Profile is a subset of the personal profile for situations where more specialised graphical interfaces are required due limitations of the display.

Daisy has been built using the MIDP profile.
Optional Packages

As different devices provide different features a number of optional libraries are provided. These include:

- Wireless Messaging API (WMA)
- Location API
- Mobile Media API (MMAPI)
- Security and trust services API
- Mobile 3D Graphics API
- Bluetooth API
- J2ME Web Services API
- File Access API

The most commonly available packages are wireless messaging, mobile media and Bluetooth. Others are currently not implemented on most phones including the Nokia 6600 being used as a test platform for this project. The location and file access API would be very useful for this application. Limitations imposed by hardware mean they cannot be used so alternative solutions to provide similar features are required.

More Information

More information on J2ME can be found on the J2ME web-site (http://java.sun.com/j2me) including specifications for CLDC, MIDP and the optional libraries. Also chapter 1 of Programming Java 2 Micro Edition on Symbian OS[14] gives a good overview of the technology.

6.3 Application Structure

When using MIDP we use MIDlets as the application framework to execute on top of CLDC. Every application must extend the MIDlet class (javax.microedition.midlet.MIDlet). MIDlets exist in the different states of the MIDlet lifecycle. These states include:

**Active:** this is when the MIDlet is functioning normally. The state is entered after the startApp() method is called. This method can be considered like the main() method of a J2SE application.

**Paused:** this state indicates that the MIDlet has been initialised but is in a dormant state.

**Destroyed:** where the MIDlet has released all resources and terminated. Entered if the destroyApp() method is called or the notifyDestroyed() method returns successfully.

This life cycle is illustrated in figure 6.3.

MIDlets will be created for the Daisy user system and the Daisy administrator system.
Listing 6.1: Using the Display

```java
// Create the display to show
Canvas new_canvas = new CanvasInterface();
Screen new_screen = new ScreenInterface();
...
// Cast the displayable object
Displayable displayable = (Displayable) new_canvas;
...
// Somewhere in the MIDlet, usually in startApp()
Display display = Display.getDisplay(this);
display.setCurrent(displayable);
```

6.4 User Interface

User interfaces are created in MIDP using LCDUI (`javax.microedition.lcdui` package). This package allows standard creation and handling of interfaces even though the displays and input available on different devices using the profile vary greatly. LCDUI is implemented individually on each device meaning it has a look and feel that is comparable to the native applications.

There are two kinds of interface available in LCDUI, a `Screen` or a `Canvas`. User interfaces are created by extending one or the other of these two base classes. Both type of interface are a subtype of `Displayable`. Any object that is a subtype of `Displayable` can be shown by a MIDlet.

In order to show any kind of interface you need control of the display. This is controlled by the MIDlet using the `Display` class. Control of the display can be passed between interfaces by passing a reference to the display when a new interface is created. The `Display` object itself is a singleton to prevent multiple threads of execution trying to manipulate the display at the same time. Use of the display is shown in listing 6.1.

User interfaces in a MIDlet are opened in turn using the `Display` object.
Only one interface can be displayed at a time. When a screen is dismissed, such as activation of a close, back or menu button it is generally desired that the previous screen should be displayed. To achieve this along with the Display object being passed to new interfaces created a reference to a Displayable object is also passed representing the screen that should be displayed next.

The architecture of LCDUI can be seen in figure 6.4.

Commands can be added to any displayable object. On series 60 phones, such as the Nokia 6600, these commands are put into a menu accessed by pressing the left soft-key. In order to handle the events triggered by the activation of commands a CommandListener must be implemented. This interface contains the method commandAction(Command, Displayable) which is called when a command is selected.

6.4.1 When to use Screen and when to use Canvas

The two most useful subtypes of Screen are Form and List. Form is used to combine different items onto a single interface. These items include image items to display images, string items to display text and choice items to display lists of elements. List is more specialised with only a list of items available.

Canvas is much more versatile than the user interfaces derived from Screen. The canvas is used to draw whatever the developer wishes using a Graphic object. Canvases are used in more graphic intensive applications such as games. Canvases have added benefits of allowing full screen display and handling of all key presses form the user.

Form, List and Canvas elements have all been used in different parts of Daisy. Each was chosen for different reasons which vary on a case by case basis.

6.4.2 Information and Error Messages

Information and error messages are presented in the Daisy system using the Alert subtype of Screen. This displays a message on a small part of the screen.
on top of the existing display. The size of the message display varies depending on how much information is displayed in the message. A timeout can be specified for the Alert display after which time the message will disappear however in most cases the messages presented in Daisy remains on screen until dismissed by the user.

There is a known issue on the Nokia 6600 when using the Alert class (issue 2.4)[15]. When using the Display.setCurrent(Alert alert, Displayable nextDisplayable) method, after the alert has been dismissed the nextDisplayable does not correctly display. To solve this problem Daisy has a wrapper for using the alert classes. This wrapper displays the alert and then catches the command to dismiss the alert so that the next display can then be correctly displayed.

6.5 Development Environment

A lot of new tools are emerging to aid J2ME developers. As Daisy is initially intended for a Nokia 6600 the Nokia Developers Suite[2] has been used in development. The NDS provides tools for creating applications and packages then deploying them to devices. Associated with the NDS are a number of emulators for different types of phone (including Symbian OS Series 40, 60 and 90). The entire suite of tools easily integrates into a number of IDEs including Eclipse [3]. This allows for fast and effective development and debugging of J2ME applications.

6.6 Deployment to Devices

When an application is ready for deployment it is packaged into a “jar” file (java archive). This archive is then deployed together with a “jad” file (java application descriptor) to the mobile device where it is installed. Applications can be deployed through a number of means including Bluetooth transfer, infrared transfer and downloading of the required files from a server.

A typical descriptor file is show in listing 6.2. MIDlets are the main executable parts of a MIDP application and are described in more detail in section 6.3.

In the past devices have had limits on the possible size of jar files. This is becoming increasingly rare and as such this problem will not be considered.

6.7 Test Platform: Nokia 6600

The Nokia 6600 is one of the new Nokia smart phones. When released it was the first mobile phone to support the MIDP 2.0 profile and MMAPI. Many other phones now follow in the direction that it has taken.

Many reviewers of the phone have identified it as having many problems with the implementation of both MIDP 2.0 and the MMAPI having many flaws. One reviewer says:

---

“...the MMAPI implementation in this phone certainly needs some serious improvements. If in doubt just take a look at the known issues document. It runs well into 15 pages and the MMAPI is not the only area that has been covered.”

6.7.1 Known Issues

There are a number of issues with Nokia 6600 MIDP 2.0 implementation. New problems with the device are being found all the time and when developing in J2ME it is important to regularly check on-line forums, especially the Nokia forum (www.forum.nokia.com) where updates on new issues are frequently posted.

Throughout implementation stages of this project the known issues caused problems with development. These issues have been identified throughout this report together with any solutions to the problems. Where problems could not be resolved details have been given of what the problem faced is and why it could not be resolved.

[10] J2ME on the 6600

Figure 6.5: Nokia Developers Suite
6.8 Device Specific Implementations

The implementations of Java vary between devices. The most obvious way that this can be seen is where optional packages are not present. It is also possible that the implementations of the provided packages may vary between devices. For example, in the Mobile Media API different encodings of media may be supported on different devices. The differences in Java implementation between devices must be dealt with in order to maximise the portability of the Daisy application.

Two ways to ensure that the java implementation in use has the required features are to either check system properties or to use a preprocessor.

6.8.1 System properties

Many details about the implementation of Java are held in the System properties. These can be queried by the programmer at run time to check that particular capabilities are present. Sample usage of system properties are given in listing [6.3](#).

Examples of properties held about a device include:

- Local device details (microedition.hostname)
- Configuration and profile in use (microedition.configuration, microedition.profiles)
Listing 6.2: Sample MIDP Java Application Descriptor

```java
MIDlet−Name: Daisy
MIDlet−Version: 1.0.1
MIDlet−Vendor: Accessibility Research Group
MicroEdition−Profile: MIDP−2.0
MicroEdition−Configuration: CLDC−1.0
MIDlet−Jar−URL: Daisy.jar
MIDlet−Jar−Size: 219694
MIDlet−Icon: /daisy_tiny_icon.png
MIDlet-1: DaisyAdmin, /daisy_logo.gif, daisy.DaisyAdmin
MIDlet-2: DaisyUser, /daisy_logo.gif, daisy.DaisyUser
```

Listing 6.3: Querying a system property

```java
import java.lang.*;

String value;
String property = "microedition.hostname";
value = System.getProperty( property );
```

- Media encodings supported (audio.encodings, video.encodings)
- Media capture support (supports.video.capture)

Unfortunately these properties give no information with respect to the attributes of how the Java implementation on the phone is constructed, only what is implemented.

### 6.8.2 Preprocessor

A preprocessor can be used when compiling the Java code for different systems. The preprocessor is used to modify source code before it is compiled. A preprocessor is not usually used with Java and is more commonly found in the C programming language. The preprocessor is not provided with the J2ME SDK. It is a separate tool that is used alongside those that are provided in the SDK. One example preprocessor for J2ME is [J2ME Polish](http://www.j2mepolish.org/).

The J2ME Polish preprocessor holds values in a device database for a variety of device specific attributes. The preprocessor can be used to insert the device specific attributes into your code when creating the application to deploy onto a specific device. Examples of the values held about devices include:

- Optional packages included in implementation (polish.JavaPackage)
- Media formats supported (polish.SoundFormat, polish.VideoFormat)
• Heap size (polish.HeapSize)
• Screen size (polish.ScreenSize)

Many of the properties held in the device database are similar to those accessible through the system properties however it does hold a wider and richer set of properties over all. The preprocessor also saves execution time as property checks are done at compile time so are not necessary at run time. More information on how J2ME Polish is used in preprocessing can be found in “The Complete Guide to J2ME Polish” [17].

Currently only the system properties and not a preprocessor have been used in ensuring that the application remains portable between devices.
Chapter 7

Architectural Issues

Before either the user or administrator system could be created several general architectural issues had to be addressed.

7.1 Storing Data on the Device

J2ME does not allow access to the local file system without the File Access API. This is currently available on very few devices, unsurprisingly it is not available on the Nokia 6600. This limitation means that storing data for Daisy using the local file system on the mobile device was never a possibility.

The only storage provided by the java implementation on mobile devices is a persistent record store. As the store is persistent any data stored on the device remains between application sessions and also after rebooting the phone.

Serialising Objects

In order to store objects in the record management store they must be serialised. To allow serialisation of objects they are given `persist` and `resurrect` methods. All objects to be stored must be given such methods.

Listing 7.1 serialises an object of type A. This is done by writing each field of the object in turn into a byte array stream. Once the data is written into the byte array stream the method returns the byte array that has been constructed. This data can then be resurrected back into the original object using the `resurrect` method as shown in listing 7.2.

Persistent Store Wrapper

In order to best handle the persistent store a wrapper was written so that any part of the Daisy system can easily write to the store and then retrieve records back with limited knowledge of how the store actually works. Multiple persistent stores can be used to store records of different types. Each record store is identified by a different name which is used when opening the store to read and write records. The wrapper does not need any prior knowledge of how
Listing 7.1: Persist code

```java
public static byte[] persist(A a)
    throws SerialisationException {
    try {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        DataOutputStream dout = new DataOutputStream(bout);

        // Write the fields of the object
        dout.writeInt(a.int_field);
        dout.writeUTF(a.string_field);

        // Return the byte array created
        return bout.toByteArray();
    }
    catch (IOException e) {
        throw new SerialisationException(
            "Could not persist object.");
    }
}
```

Listing 7.2: Resurrect code

```java
public static A resurrect(byte[] data)
    throws SerialisationException {
    try {
        ByteArrayInputStream bin = new ByteArrayInputStream(data);
        DataInputStream din = new DataInputStream(bin);

        // Create the new object
        A a = new A();

        // Read and set the fields of the object
        a.int_field = din.readInt();
        a.string_field = din.readUTF();

        // Return the object created
        return a;
    }
    catch (Exception e) {
        throw new SerialisationException(
            "Could not resurrect object.");
    }
}
```
many stores are used or what their names are. The name of the store is passed in as a parameter to store methods.

The core methods provided by the persistent store wrapper include:

\begin{verbatim}
int store(byte[], String): stores byte data to a record in the named store, returns the record number.
int update(byte[], int, String): updates a record specified by a record number in the named store, returns the record number.
byte[] loadRecord(byte[], int, String): loads the specified record from the named store and returns the byte data.
void removeRecord(int, String): removes the record with the specified record number from the named store.
void clear(String): removes all records from the named store.
\end{verbatim}

As well as the wrapper to perform standard store operations an enumeration wrapper has been created so that it is possible to get an enumeration over the persistent store of a particular kind of records without knowing how the implementation of the record management store works. The enumeration over the persistent store is based on the Enumeration class provided by the Java language, methods available include:

\begin{verbatim}
PersistentStoreEnumerator(String): constructor creates a enumeration over the named store.
boolean hasMoreElements(): returns true iff there are more elements in the enumeration.
byte[] getNextRecord(): returns the byte data for the next record in the store.
byte[] getNextRecordId(): returns the record number for the next record in the store.
void close(): closes the enumeration over the persistent store.
\end{verbatim}

**Record Size**

Records have a maximum size of 128Kb on the Nokia 6600. In order to ensure that records can be stored even if they exceed this size the before being put into the persistent store data is split into storable chunks. The start of each record in the record management system contains the length of the data stored in the record and whether it is the start of a record or a continuation of another record. Whenever data is split across multiple records each record holds a pointer to the next so that the original record can easily be reconstructed. The record number to represent the whole stored record points to the record holding the first chunk of the original data.

A problem with this method is that the maximum record size varies between devices. Currently this is set as a constant at compile time. It cannot be
dynamically discovered when deploying the application to different devices as no system property holds this value. This means that without changing the constant in the source code this application cannot be deployed to devices with less than 128Kb record size. Similarly it will not make full use of the potential of any device with a greater record size. The records will still be split creating an unnecessary overhead on when storing data.

In order to change the constant for different devices the source would have to be modified or different classes used for different devices each with a different value for the constant. A preprocessor such as J2ME Polish as mentioned in section 6.8.2 could be used to insert the correct value of the constant and automatically produce binary files suitable for different devices however currently a preprocessor is not being used in development of this project.

Managing the Persistent Store

A utility to show the number of records contained in each persistent store has been created which will be included in the administrator system. This allows the administrator to see how many individual records have been created in each store, clear the individual stores and also see the total amount of memory available to expand the record stores.

![Store Management Interface](image)

Figure 7.1: Store Management Interface

7.2 Settings

A number of different settings are required throughout the system. The settings must be stored between sessions of the application. To achieve this the settings are stored in the persistent store and accessed using a settings library. When the settings library is requested to fetch a particular setting it either returns a value from the persistent store, the default setting value if no stored value exists or null if the setting requested has not been defined.

Settings can have different types including:

- String
- Integer
- Bluetooth Device
Switch (either ON or OFF\textsuperscript{1})

URL

Settings are fetched using static calls to the `Settings` class as illustrated in listing 7.3. The setting value must be cast to an appropriate type depending on the type of the setting that was requested. Similarly settings are set using another static call as shown in listing 7.4.

The calls must be made with a valid setting name. Some settings that have been defined include:

- “Logging”: switch which is either ON or OFF
- “GPSDevice”: Bluetooth device representing device used in tracking
- “Username” string to describe the user
- “ProximityThreshold” integer for location finding

The administrator system includes a settings modification tool that displays the full list of settings. Each setting can be modified by selecting it from the list. Once selected a suitable interface is used to modify the setting depending on the type of the setting. For example, a string setting is modified using a text box whereas a Bluetooth Device setting is modified by using the Bluetooth tool written to search for a local Bluetooth device.

### 7.2.1 Bluetooth Device Discovery

Bluetooth operates by one device offering a service to which other devices can connect. A device wishing to offer a service creates a service record which it registers with its locally and then waits for connections. A querying device searches for all devices in range that are set in discoverable mode. Once a device is found, the querying device checks that the found device offers the required service and then connects to this service.

\textsuperscript{1}This is very similar to a boolean however it is not referred to as a boolean to avoid confusion. The switch is stored as an integer type using constant values representing ON and OFF rather than using the Java boolean type.
To discover a Bluetooth device for use in Daisy a Bluetooth device discovery utility has been provided. This is used with the settings tools to provide a value for any settings of the type “Bluetooth Device”.

To find a Bluetooth device to communicate with the java `DiscoveryListener` interface must be implemented. This is used with a `DiscoveryAgent` which is initialised using the local Bluetooth device. The agent communicates with the local device, searching for new devices, sending the results back to the discovery listener. When the agent has finished searching for devices it informs the discovery listener.

The utility to discover nearby Bluetooth devices uses a `BtDeviceDiscoverer` that implements the `DiscoveryListener` interface and controls the discovery agent. The discovery process is started from a `List` user interface and as results
come in from the discovery agent the device discoverer sends these results to be
displayed on the list interface. Once the search is complete the user may select
one of the found devices from the list. The selected device is then returned
to another object which is listening for the device by implementing the Daisy
BtDeviceListener interface.

7.3 Media

All media is stored in a media library. The media library is used to store
individual pieces of media locally and retrieve them when needed. The media
library is capable of retrieving remote media or returning a default NO_MEDIA
value when the required media does not exist in the media library.

The media used in Daisy comes in three flavours:

- Image media
- Audio media
- Video media

A standard media class exists which performs functions that are common to
all media types. The common functionality includes:

- Getting an image or thumbnail to represent the media.
- Creating a display to be put into a form that will give a visual representa-
tion of the media.
- Serialising (persisting)/resurrecting the media to/from byte data.

By extending the Media class each different type of media can be treated in
the same way by the media library. The major differences between the media
types are in how they are presented to the user. An image is simply drawn on
the screen, audio requires a player but has no visual display, video combines
both of these methods needing showing on the screen and a player to control
it. To best control media requiring a player a MediaPlayer interface has been
provided that allows a player to be created, started and stopped for a particular
item of media.

The media library is a singleton which ensures that whenever the media
library is used it is always the same instance. This minimises the number of
times media is loaded from the persistent store. When media is required in the
Daisy system, the media library instance is retrieved and the instance of the
library is used to get whatever media is needed.

Retrieving media

When media is requested from the library it is searched for in two ways. First
of all the system checks to see if it exists in the local library. If this fails the
next step is to search for the media at a remote site. The remote site is defined
in the settings using the “MediaPath” setting. This path is used to fetch media
using HTTP from a server holding the required route media. A remote site is only used if the “UseRemoteMedia” switch setting is set to ON.

Media is fetched from the remote site using HTTP over GPRS. This requires network connectivity. If the media cannot be fetched from the remote site the default NO_MEDIA item is used instead to indicate that the media could not be retrieved.

Two items are included with the deployed application. These are retrieved locally by using the getResourceAsStream(String) method call in the class Class class. The items stored as resources in the application archive include:

**No media image:** used when media cannot be retrieved locally or from a remote resource. This is the default media item and is required to be a placeholder when no other media exists.

**Audio image:** used as a pictorial representation when a piece of media is audio. Otherwise there would be no visual display to indicate that the media has been loaded.

**Storing media**

When a new piece of media is retrieved from a remote site it is put into the media library. This involves storing the media locally in the persistent store. By keeping a local copy of the media the next time it is needed it can easily be retrieved without going to the remote site.

**Playable Media**

Audio and Video media are forms of playable media. They implement the Daisy interface MediaPlayer. This means that they must provide the core set of controls needed to play active media. These methods implemented include:

- `createPlayer()`
- `startPlayer()`
• stopPlayer()
• closePlayer()

The media players for audio and video work by creating a Player object as defined in the Mobile Media API. This object is created using a data stream of the byte data associated with the media item. When the player is started it blocks the rest of the application unless it is run in a separate thread. Media players create threads that play the media enabling the user to continue to control the Daisy interface while it plays. Threads are created by implementing the Runnable interface.

7.4 Routes

Routes are represented as a list of route nodes. They provide step by step instructions from an origin node through to a destination node. Each route node consists of:

• Title for the node (e.g. Tesco)
• Description of the node (e.g. You have arrived)
• Media associated with the node.
• Location of the node (see section 7.5)

The route nodes are connected in a linked list accessed through a Route object. This route object holds a reference to the first node in the route and subsequently each route node then points to it successor. At any time an instance of a route object also maintains a pointer to a current route node representing the route node that is being viewed at that time. The route structure allows you to move around the linked list with access operations including:

• getCurrentNode()
• getFirstNode()
• getNextNode()
• getPreviousNode()
• getLastNode()
• getNodeN(int) - get the Nth node in the route

These operations mean it is possible to move backwards and forwards through the route as well as moving straight to specific nodes.
7.4.1 Limitation of Route Structure

A limitation of this route structure is that it only allows for a single path through the route. There is no scope for varying the designated route and still arriving at the destination. This means that should the user wander of the proposed route or find an obstacle in their path then the Daisy system is not likely to offer them much support in finding their way back on track.

A better way of specifying routes would be to have a much larger collection of route nodes from which a route to a particular destination can be picked out.

An example of a simple route is shown in figure 7.6, each dot represents a route node at the intersection of paths. This shows a route as a single list of route nodes. There is only one path through the system so if the user takes one of the alternative paths by mistake they will not be able to find their way to their required destination.

![Figure 7.6: Diagram of an example simple list route](image)

More complex routes require more route nodes like that shown in figure 7.7. These more complex routes mean that if a user takes an incorrect path, Daisy would be able to recalculate their route to the destination. A large store of route nodes would be needed so that if the user strays from their route a new route can easily be calculated.

A simple solution to add correction to routes would be to keep the existing route structure where there is a single route from start to finish but add some surrounding route nodes to complement it. These surrounding route nodes would provide the user directions back onto the correct route should they get lost. A mesh of route nodes like this would cope with a user taking some expected wrong turns. This method has been illustrated in figure 7.8.

The problem with specifying a route in this way would be in knowing which additional route nodes should be provided to get a user back on track. If there are a many different incorrect paths at each route node the a large number of
Figure 7.7: Diagram of an example complex mesh of routes

additional route nodes are needed.

Figure 7.8: Diagram of an example mesh route using the simple solution

7.4.2 Reasons for choice of Route Structure

Routes have been implemented using the linked list of route nodes to provide a single path. This has the potential to be extended to at least include the simple solution. To create a more rich store of route nodes that can be used to create dynamic routes will require more sophisticated path finding algorithms which
is not at this stage a priority for Daisy.

It is not simple to specify many different route nodes. The reason for this is that Daisy routes are media rich and require very precise directions from one node to another. The routes specified are very carefully planned so that they are easy for the users to follow. If a large number surrounding route nodes were required, with multiple directions out of each a lot of media data would have to be captured and stored. Acquisition of such large amounts of data is not feasible for travel trainers creating the Daisy routes.

### 7.5 Locations

The locations of each route node are used in tracking. Locations use the longitude and latitude of a point. These locations are found using a GPS device connected to the Daisy system.

J2ME has a Location API as an optional package. Unfortunately this package is not included in the Nokia 6600, in fact, it not supported in any CLDC 1.0 phones as it requires floating point support which is only provided in CLDC 1.1 (see section 6.2.1 for more information on J2ME configurations). This limitation meant that location classes were needed to be produced specifically for Daisy.

Location objects are used to hold the longitude and latitude of a particular location. Latitude gives the elevation from the equator and longitude the angle of rotation from the Greenwich meridian. Each of the longitude and latitude values consist of an integer degrees value, a floating point value for the minutes (see section 7.5.3 on floating point numbers) and a direction. Seconds are included as the fractional part of the minutes value. The directions are north or south for latitude values and east or west for longitude values.

#### 7.5.1 Location Discovery

The location information is discovered from a GPS device. GPS systems are described in detail in section 2.3.2. The GPS device sends out location information via Bluetooth using the NMEA-0183 protocol (section 2.3.2). In order to extract the location from the stream of NMEA-0183 data a parser has been created.

When created the parser is given a stream from which the NMEA-0183 data can be read. The parser runs as a thread so that is can continuously read the stream of information and update its location while the main thread of execution continues to interface with the user and run other aspects of the Daisy system.

The parser holds a reference to an NMEALeanser object. When the parser discovers new information it reports this back to the listener. NMEALeanser is an interface that any class wishing to use the parser to get information from a NMEA-0183 data stream may implement. The reference to the listener is passed to the parser when it is created. The listener interface includes methods to set:

- Location (setLocation(Location))
- Number of satellites visible (setSatellites(int))
• Fix - whether or not the GPS has a location fix (`setFix(boolean)`)
• Proximity - distance in meters of last read location value to last reported location (`setProximity(long)`)
• Raw data - string of raw NMEA-0183 data (`setRaw(String)`)

Most of these are only used in the GPS display interface. This interface was again designed as a utility for the administrator system. It shows the current state of a connected GPS. This will be used in collecting location details at particular locations and determining how strong a signal is available at different locations.

![GPS Status](image)

Figure 7.9: GPS status interface with and without a location fix

### 7.5.2 Location Tracking

One of the major parts of the Daisy system is the ability to connect the mobile phone running Daisy to a GPS in order to track the users progression through the route. A Bluetooth GPS device must be located and stored in the settings for the “GPSDevice” setting to be used for tracking. The device is discovered as described in section 7.2.1.

A Tracker object has been created to control the communication with the GPS device. The tracker is a singleton to ensure that if tracking is used in different parts of the system the same tracker is always used. Another listener interface has been so that any classes wishing to use a tracker to get location information need only implement the `TrackerListener` interface and the register themselves with the tracker.

The tracker provides several different services to the Daisy system including:

- Handling of GPS streams over Bluetooth
- Communication with the NMEA parser
- Connection of multiple `TrackerListener` objects to the same tracker.
- Ability to start and stop the tracker.
- Ability to keep the tracker running but turn off reporting to listeners.
There were some problems with the robustness of the tracker. Although it is possible to connect to and then disconnect from the tracker if has been found that if you disconnect from a GPS device you cannot reconnect to the same device without restarting the application. This causes issues as the users may wish to stop tracking intermittently.

The solution to this problem was to leave the tracker running at all times once it has started. Only when the application closes or a new device is specified to be tracked does the tracker stop parsing data from the GPS device. Instead of stopping the tracker other parts of the Daisy system instead remove their listeners from the tracker. The tracker continues to run in the background thread but with no other objects to report locations back to. As it is a singleton future object wishing to use tracking can easily find the instance of the tracker and register their own listeners.

**Location Proximity**

In order to find the users position in a particular route it is necessary to perform location proximity calculations. These will be used while tracking to find the closest route node to the users current location and know when to change the route node being displayed to the user.

It has been decided that proximity values will only go down as far as meter accuracy as anything less than this is not necessary as we are representing the movement of people. Also given that the precision of the GPS systems does not go down to such a fine granularity any greater accuracy would be wasteful. This means that proximity values will be represented as integers which immediately gives the advantage that the values can be manipulated using J2ME CLDC 1.0 which only supports integer and not floating point arithmetic.

There are various different methods of finding the proximity between two locations specified by longitude and latitude co-ordinates. Below are two approximations followed by a more accurate method of finding the distance. The calculations are given in more detail in the paper “Geographical Distance Calculations” [18]. Models use kilometers as units which can easily be converted to meters as required for our proximity calculations.

The equations use the following values:

- $d_{ist}$ the distance in km between the two points
- $d_{ist_{long}}$ the longitude distance in km between the two points
- $d_{ist_{lat}}$ the latitude distance in km between the two points
- $long_1$ the longitude of the first point
- $lat_1$ the latitude of the first point
- $long_2$ the longitude of the second point
- $lat_2$ the latitude of the second point
Approximate Formula 1

This method uses very basic trigonometry assuming that 1 longitude degree is 111.2km and 1 latitude degree is 85.3km. These are only assumptions as the earth is not a perfect sphere and the radius decreases towards the poles.

\[
\begin{align*}
\text{dist}_{\text{lat}} &= 111.2 (\text{lat}_2 - \text{lat}_1) \\
\text{dist}_{\text{long}} &= 85.3 (\text{long}_2 - \text{long}_1) \\
\text{proximity} &= \sqrt{\text{dist}_{\text{lat}}^2 + \text{dist}_{\text{long}}^2}
\end{align*}
\] (7.1)

Approximate Formula 2

This second approximation builds on the first by calculating the distance between the longitude degrees using the position at the edge of the sphere.

\[
\begin{align*}
\text{dist}_{\text{lat}} &= 111.2 (\text{lat}_2 - \text{lat}_1) \\
\text{dist}_{\text{long}} &= 111.2 (\text{long}_2 - \text{long}_1) \cos(\text{lat}_1/92.2) \\
\text{proximity} &= \sqrt{\text{dist}_{\text{lat}}^2 + \text{dist}_{\text{long}}^2}
\end{align*}
\] (7.2)

Great Distance Formula

Using spherical geometry it is possible to find the precise distance between points.

\[
\begin{align*}
\sinpart &= \sin(\text{lat}_1) \sin(\text{lat}_2) \\
\cospart &= \cos(\text{lat}_1) \cos(\text{lat}_2) \cos(\text{long}_2 - \text{long}_1) \\
\text{proximity} &= 111.2 \left(\frac{180}{\pi}\right) \arccos(\sinpart + \cospart)
\end{align*}
\] (7.3)

After performing some analysis on these equations it has been found that the accuracy of the Greatest Distance and Approximation 2 are fairly close. Results using each formula for a pair of points near Imperial College London (figure 7.10) are shown in table 7.1.

The exact distance between the points is unknown however it is known to be around 50m indicating that the greatest distance formula and approximation 2 are correct. This calculation demonstrates how there is very little difference between the two results and for our purposes either would be suitable.

Approximation 1 clearly gives a much less accurate proximity than the other two values. If the greatest distance value is taken to be correct the percentage error of approximation 1 is 21.9%. All calculations using co-ordinates from the London area were found to give similar errors. It would be possible to correct for

57
Table 7.1: Sample Results using Proximity Formulas

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latitude 1</td>
<td>51 degrees 29 minutes 58.92 seconds</td>
</tr>
<tr>
<td>Longitude 1</td>
<td>0 degrees 10 minutes 44.40 seconds</td>
</tr>
<tr>
<td>Latitude 2</td>
<td>51 degrees 29 minutes 59.34 seconds</td>
</tr>
<tr>
<td>Longitude 2</td>
<td>0 degrees 10 minutes 46.92 seconds</td>
</tr>
<tr>
<td>Approximation 1</td>
<td>61.09m</td>
</tr>
<tr>
<td>Approximation 2</td>
<td>50.14m</td>
</tr>
<tr>
<td>Greatest Distance Formula</td>
<td>50.11m</td>
</tr>
</tbody>
</table>

this error assuming the system will be used in the same area however it would be better to use one of the other formulas to make the system more general.

When deciding between the second approximation and greatest distance model the simpler approximation was chosen. As both deliver comparable results the simpler method was chosen to reduce the computation required.

A fundamental problem with using any of these models is caused by the limitations found with J2ME in that floating point numbers are not supported by all configurations. Floating point numbers are required in order to give a suitable level of precision. Integer arithmetic alone would not let us resolve distances down to the meter level as required. This limitation prevents us from using any of the models until some method of performing floating point computation is made available.

7.5.3 Floating Point Numbers

The J2ME configuration CLDC 1.0 which is used on the Nokia 6600 along with many other mobile phones does not support floating point numbers. In order to get around this restriction there are a several alternatives:

Upgrade to a different configuration: there are configurations for J2ME that support floating point arithmetic (e.g. CLDC 1.1). Using these instead of the current configuration would mean that the devices used
would have to be changed as the version and configuration of Java on a device cannot be changed. As we are interested in using Daisy on a variety of handsets widely available today including the Nokia 6600 this limitation on which hardware can be used is not desirable.

**Use only integer arithmetic:** this removes the need for floating point arithmetic altogether. This is not a suitable solution as it would not be possible to produce accurate proximity calculations.

**Use fixed point arithmetic:** this method allows you to store numbers with a fixed number of digits after the decimal point. Again fixed point numbers are not provided by J2ME CLDC 1.0 and must be added separately. Fixed point numbers have advantages as they can be optimised to work faster than floating point numbers and can also represent some values more accurately than floating point numbers. An example of the increased accuracy is seen when storing a value such as 0.1. In fixed point arithmetic 0.1 is stored where as it is approximated in floating point representation as the floating point number is stored base 2.

Fixed Point arithmetic is already used in many gaming applications where it is necessary to optimise calculations, such as when using 3D or other high quality graphics. Fixed point arithmetic will not be used in Daisy because of the complexity involved in creating fixed point numbers with all the necessary functions included when suitable floating point representations are freely available.

**Use a floating point implementation:** Several implementations of floating point numbers have been developed using the `long` data type to represent the mantissa and exponent that form the floating point numbers. Such classes also provide various functions to perform calculations such as trigonometry that will be required in the location calculations.

Having decided to use an imported floating point package with Daisy the next decision was to decide which should be used. Two suitable versions were considered. To ensure that they provided suitable accuracy sample location calculations were performed using the classes. The classes used in the evaluation included:

- `henson.midp.Float` (developed by Nikolay Klimchuk)
- `ral.Real` (developed by Roar Lauritzen)
- `java.lang.double` (available in J2SE but not J2ME)

The `java.lang.double` class from J2SE was used to evaluate the accuracy of the other two implementations of floating point numbers. The A sample calculation is given here calculated using each of the three classes. The same location point has been used in this sample as used in table 7.1.

The evaluation results are show in table 7.2. It was found that the different implementations were mostly accurate. The only interesting result was when

---

2[http://henson.newmail.ru/j2me/Float.htm](http://henson.newmail.ru/j2me/Float.htm)
3[http://gridbug.ods.org/Real.html](http://gridbug.ods.org/Real.html)
using the greatest distance formula with the `henson.midp.Float` class. The class is not accurate when using the greatest distance formula. This is because the “acos” function provided with the class do not work correctly with very small values. Otherwise all functions used from this class operated correctly. This means that although it cannot be used for greatest distance calculations it can be used for calculations involving approximations 1 and 2.

Having established accuracy of the classes and having found that the performance of each are comparable the decision on which class to use came down to ease of use. The `henson.midp.Float` class was found to be far easier to use. One reason for this is that it has an API more closely matched to those existing in the Java language. Further more tools are provided with the class to convert directly from using the Java type `double` to using the new class. This came in very useful for verifying the correctness of the implementation of the formulae.

### 7.6 Logging

When developing with mobile devices there is no console output. Anything written to `System.out` is simply discarded. This means that it is difficult to log and debug programs. Problems particularly arise when exceptions are raised. If not handled correctly the exceptions will simply cause the MIDlet to close without any indication of what the problem was that caused the issue.

To solve this problem a logging system was provided for Daisy. The logging system stores logged messages into the persistent store. This means that if the application fails the log still exists so that problems can be diagnosed after they happen. Logged messages are stored together with a time stamp in seconds since the log was last cleared. Each time the Daisy system is started the log is cleared.

The logging system runs in a separate thread. This means that logging operations do not interfere with the execution of the main system. If essential logging operations are needed it is possible to ensure that messages sent to the logging thread are written to the store before execution of the main thread continues.

The logging system is the only part of Daisy that handles its own persistent store. This is done so that the record store for the log can be kept open whenever the thread is running. This approach reduces the number of operations required in logging messages as the store does not have to be continuously opened and closed whenever a message is to be logged. Unlike other sections of Daisy using the persistent store logging frequently accesses so the overhead of opening and closing the store would otherwise be substantial.

A logging GUI has been provided that shows all of the messages in the log.
It also provides a facility to clear the log if the user wishes. The logging GUI is only accessible through the administrator system.

Figure 7.11:  Logging GUI
Chapter 8

Daisy User System: Stage 1

The Daisy user system is the part of the overall Daisy system that the users with learning difficulties will use. It contains a minimum number of features so that it is suitable for them. The basic functionality of Daisy is provided allowing route selection and route display. Within route display GPS tracking can be enabled so that a route can be followed with instructions given to the user when necessary.

![Daisy User System Main Screen](image)

Figure 8.1: Daisy user system main screen

8.1 Route selection

Daisy users will make use of the system to follow different routes depending on where they wish to go. As they will have different destinations that they desire to reach it must be possible to select a route from a number of different stored routes. A route selection interface has been provided that is accessible from the main Daisy user screen.

Routes are loaded from the persistent store to be displayed on the selection interface. As only one operation is required from the route selection interface, which is to choose an item from the list of possible routes, a List interface has been chosen for this purpose. The routes are loaded in turn from the persistent store and the name is put into the list ready for selection.

When a route is selected from the list the entire route is loaded from the persistent store. The “Route” setting is set to the name of the current route so
that when the application is closed the selected route will be remembered when the application is subsequently reopened. After a route has been selected the next interface displayed is the route display list.

### 8.1.1 Remote Routes

In addition to the routes stored locally it is also possible to load in remote routes. In the settings a “RoutePath” URL is held which is the location of a server containing an additional store of route data. Another setting “UseRemoteRoutes” specifies whether or not this store should be accessed when loading routes for selection.

If the remote routes are to be used Daisy attempts to get a "routes.dat" file from the specified server. This file contains the names of remotely stored routes and the locations of their route data. The names are added into the route selection list ready to be selected. If the remote route file cannot be retrieved only local routes can be selected.

If a remote route is selected the next task is to get the remote route data. This is done using the location of the route file which was downloaded with the names of the routes. The system attempts to retrieve the remote route file from the server. If it cannot the system displays a warning and returns to the main interface.

Route files contain multiple lines of the form:

```
Node title,5124.6437,N,01105.8707,W,Description,Media name
```

That is each file contains a line for each route node in the route. The line consists of the title, location (latitude and longitude), a description of the node and the media associated with the node. The media used in the remote route may already be present on the device or it may also need to be retrieved from a remote source (see section 7.3 for information on remote media). The route files are parsed by Daisy and used to create a route object to be used in the application.
8.2 Route display

Route display is only enabled in the user system when a route is loaded. A route is either loaded at startup from the persistent store using the setting "Route" that specifies the currently selected route or by selecting a route with the route selection interface. When a route is loaded the main Daisy user system interface shows the name of the route that is loaded as in figure 8.5.

![Daisy user system with a selected route](image)

Figure 8.3: Daisy user system with a selected route

There are two ways that a loaded route can be displayed, either using a list of all the nodes or using a node per screen view. It is important to remember at this stage that the route is given to provide two forms of information: pre-trip planning and in-trip reassurance (see section 2.1.3).

At this stage only image media is available in routes, audio and video media will be added in stage 2 of the user system development. This was intended to simplify development so that the principle behind how routes are displayed could be established before expanding the system to include more complicated media.

8.2.1 Route node list

The first method of displaying the route is by giving a list containing all of the route nodes in order. This is put into another List interface. Each element in the list specifies one route node. The elements of the list consist of an image to represent the route node which is the image media associated with the node and the title given to the node. The route list is shown in figure 8.4.

![Route listing interface](image)

Figure 8.4: Route listing interface
The list of nodes allows the user pre-trip planning. By looking through the list of nodes they can easily see each step of the route grouped together. This may help by instructing them on how many landmarks they need to pass through, and to give them an idea of the landmarks that they will expect to see.

The list does not allow users to view the images associated with route nodes in detail. In order to get more detail for each route node the users can select any of the route nodes from the list. This will take them into the single route node display.

8.2.2 Single route node display

The single route node display shows the image associated with the route node much larger than it is displayed on the route list. This means that the user will see a lot more detail and be able to accurately ascertain what landmark the image represents. In addition to the image displayed the display also gives the title of the node and the description associated with it.

As the single node display contains a number of different pieces of information it was chosen to be displayed on a Form user interface. This allows each of the different items, in this case an image and a description to be easily added onto the display. A route node display can be seen in figure 8.2.

![Figure 8.5: Route node display interface](image)

The route node display provides both pre-trip planning and in-trip reassurance.

Pre-trip planning is provided as the user can look at each node in detail to familiarise themselves with the overall route. Commands associated with the display allow the user to view the next and previous route nodes in the route easily so that they can examine the route from start to finish looking at the points that they will need to pass through to arrive at their destination.

Using the same operations in-trip reassurance can be established. The user can manually follow themselves along the route, using the “Next Route Node” command to track themselves as they move through the route.

More sophisticated in-trip reassurance is supplied when the route display is connected to a tracker (for more information on trackers see section 7.5.2). By implementing the interface TrackerListener and registering itself with the tracker provided by Daisy the route node display can receive notifications of when the users location changes. This notification can be used to compute the
closest route node to the users location and display the correct route node. As the user moves the tracking system automatically follows their progress and changes the route node display to give the user instructions based on their current location.

The default mode is with tracking switched off. The user can easily connect a tracker by selecting the “GPS Tracking” command from the menu.
Chapter 9

User Meeting and Feedback

One of the key elements of the Daisy system is that it is suitable for the intended users. The Accessibility Research Group from UCL who requested the system agreed to come to Imperial to test the system after the first stage of user system development. It was not suitable for the end users, those with learning difficulties, to trial the system as they would not be able to give suitable feedback to aid development. Having extensive experience dealing with the end users the members of the research group were able to comment on their behalf.

At the time of demonstration only a prototype of the user system was ready. The administrator system was not discussed during the meeting as it is less important as it will not be used by the users with learning disabilities.

The demonstration involved a walking tour around the streets around Imperial College. The prototype of the user system was used in the demonstration to show the functionality midway through development. A sample route had been pre-programmed into the system ready for the demonstration. The route consisted of 8 route points, each with a photograph, name and description. At this stage no other media could be displayed by Daisy.

![Figure 9.1: A sample route node display](image)

GPS tracking was used in order to guide the group throughout the demonstration. The images shown and descriptions given by Daisy were used to determine which direction should be taken. As the group followed the route display changed according to their location as required.

Two members of the research group attended the demonstration and follow-
The test run of the system they commented on the features available. The discussion focussed on:

- The current operation of the system from a users perspective
- How the current system could be improved to better suit the users
- Additional features that would benefit users.

9.1 Comments on Current Operation

Overall members of the research group were satisfied by the current level of development. They found the Daisy environment to be a suitable solution to the problem in hand however there were a number of areas that they felt could be improved upon.

The main changes requested were in the way that the routes are described to the user. Additional instructions are required rather than just specifying simple directions. More work should also be put into how the location changes when the user moves between route points.

It was noted that at some stages the information displayed to the users may not have been appropriate sometimes with too much information presented at once or not enough emphasis put onto the more important information shown.

The methods of input to the system were adequate without too many options presented in menus however at times it seemed that the method of control required overly complicated access to menus rather than more intuitive activation of functions from the keyboard. The joystick and other keys on the keypad were found to not be utilised as much as they would have liked them to have been.

9.2 Possible Improvements

9.2.1 Route Node Display

Consider for example the following display including a title, and image and a description.

Figure 9.2: Route node display of Queensgate
The research group specified that the users gain most benefit from the image presented to them rather than either the title or the description. The well known saying "a picture tells a thousand words" comes to mind and is especially relevant for the users in this situation who would find it extremely difficult to determine their location without the assistance of the image.

Ideally the image should cover a lot more of the available screen area making it easy to interpret and use to pull out key information. This would involve reducing the space allocated to the title bar and description as much as possible and to increase the size of the image. This would have the added benefit of reducing the number of sources of information presented at once and hence reducing the potential for confusion.

9.2.2 Route Selection

In the prototype in order to select a possible route to follow a list of routes is presented in textual form. This is not suitable for the users as they will struggle to recognise a route based on purely a textual description.

![Figure 9.3: Route selection GUI](image)

A far better way for the users to identify a route is by showing an image that they can easily associate with the route. This could be a recognisable landmark along the route or more likely the destination.

The function to select from a number of routes was otherwise seen to be very useful by the research group as it gives additional freedoms to the users by allowing them increased choice over what they do.

9.2.3 Give both directions and a display media to the user

At each route node in order to assist the user as best possible the research group suggested two pieces of media be presented. In addition to the display media that shows the next target point another piece of media should be provided giving directions to the location specified by the route node from the previous route node.

The direction media should be able to be viewed/played as many times as necessary but dismissed when not needed in order to show the media describing the target. It is most likely that a route node will display an image to allow orientation and have a piece of audio with it to give instructions.
9.2.4 Tracking Mode

Currently in tracking mode the system automatically moves between instructions when you approach a particular location. It is desired that a better approach should be for the user to move to a specific location, at which point they would be notified that they have reached the location. The instructions would be used in order to guide them to this location.

Once in the correct position the user can use the image provided by the system for orientation. Once they have got their orientation right they will be able to signal to the system that they are ready for the next instruction to be given. The tracking mode will be used to ensure that the user is moving in the correct direction rather than automatically scrolling between the different route node points.

In addition to this modification to how the tracking mode works it is desired that the GPS tracking should be made more stable. It was found that it was possible for the tracker to switch between adjacent route nodes based on fluctuations in the GPS readings rather than movement of the user. The diagram below attempts to explain what happens in this situation.

With a user at point x midway between locations A and B, fluctuations in the GPS signal can cause the daisy system to identify the user as moving closer to one point than the other when the user never moved and the display should not have changed. The tracker currently copes with only registering movements over a certain threshold value from where the last location reading was taken (say every 1m) but random fluctuations in signal, reporting incorrect locations, can lead to wrong results being displayed. In order to counter this problem the tracker should only report a change in location after a consistent signal is received at the new location in order to avoid poor signal variation.

9.3 Additional Features

9.3.1 Assistance Feature

As mentioned in the specification an assistance feature is desirable in the end Daisy system. This feature was not present in the prototype system and is now considered to be a higher priority by the research group than originally stated. The assistance request should be sent from the phone via text message so that a helper can then contact the user requiring assistance.
9.3.2 Notification of arrival at particular points on route

The user should be notified of their arrival at a particular point on the route, i.e. when information displayed to them by Daisy changes. This is an important feature as the user may have put the phone into their pocket or be unaware that they have reached the particular destination point. Notification should be made through either audio or vibration as the user will not necessarily be viewing the screen at the time of notification.

9.3.3 Back-light always on

Periodically the back-light on the mobile phone screen switches itself off in order to preserve battery life. The research group indicated that this was undesirable for Daisy as often the users can take some time to identify their location by looking at the image on the screen and when the light turns off the image becomes difficult to view. Although the light will come back on when the keypad is touched the users may not be aware of this so would not know how to reactivate the back-light.

9.4 Outcome from the demonstration

The discussion following the demonstration resulted in some changes to the original specification. As these changes were not seen to create major upheavals with the system structure they will all be taken into consideration in further development. The changes should not affect the progress of the project as it was anticipated that there would be a requirement for some alterations based following the meeting.

9.4.1 Key concerns identified during meeting

Based on the discussion the following key tasks have been identified for development:

1. Route node structure change: orientation (image) and instruction media (audio) rather than just a single image.

2. Route selection menu should incorporate an image associated with the possible routes.

3. Route node display does not need a lot of information present, this should be removed to emphasise the orientation media. The display should be made full screen to maximise the area of the screen used to display useful information.

4. Tracking method should wait for the user to confirm arrival at each route node before changing the display. Also the tracker should be adjusted to make sure changes in signal are not just fluctuations.

5. Assistance feature needed to be incorporated into the user system.

7. Leave the back-light on at all times to make it easier to see the screen.

8. Make use of joystick controls in order to make input into the system more intuitive for the users.
Chapter 10

Daisy User System: Stage 2

Following the meeting with the users concerning the development of Daisy several improvements were made. There were also a number of limitations that prevented suggested changes being made. The changes are laid out in this section.

10.1 Route node media

Up until now only image media has been supported. Support for Audio and Video media were included in the second stage of development. Details of how these are implemented can be seen in section 7.3.

A change to the media associated with a route node has been made. Before each node had only one piece of media. It now has:

**Orientation media** is an image that the user can line up with their surroundings at a particular location. This orientates the user as they will be facing in the correct direction to continue on the route.

**Instruction media** is an audio clip used to give the user instructions from the last location that they were last known to be in (at previous route node) to the next route node point where they can position themselves with the orientation media. Audio media is used to describe what they need to do next.

The orientation media display is simple as it is just the same as before with the image shown on the screen.

The instructions media is played in two ways. Firstly it is played when the user first views the route node. The player is started automatically to give the instructions to the user. The user can repeat the instructions at any time using a control on the route node display interface.

10.2 Route selection by image

Currently route selection is done using a list of names for routes. This has been deemed inappropriate as the users will find it difficult to recognise the route
from its name and would find it much more useful to be given an image to associate with the route. The problem is described in section 9.2.2.

The image chosen to represent the route is the image (orientation media) associated with the last node in the route. This image has been chosen because it will be an image of the destination node. This is the landmark most likely to be recognisable for the route as a whole. It also make logical sense as then the users will be able to choose their route based on the name of the destination and also a picture of that destination.

Figure 10.1: Route selection by image

The List interface used to select the route has been expanded to give images of the possible routes in addition to the names of the routes. This is easy for locally stored routes as you can simply load the route from the persistent store and get the last media image. It is more complicated for remote routes as the entire route cannot be loaded to find the media for the last node as this would require a lot of data transfer. To solve the problem a media name was added to the route description file routes.dat retrieved from the server. This file now contains names of remotely stored routes, locations of their route data and the name of a piece of media associated with the route. The media specified could be either local or remote media.

10.3 Improving route node display

During the demonstration it was thought that too much information was displayed on screen when showing each route node. The old interface is shown in figure 8.5.

In an attempt to de-clutter the interface the description has been removed as it was seen to add little value. The description can now be viewed in a message box rather than always being on the screen (figure 10.2).

By creating the interface using a canvas rather than a form it is becomes possible to remove the title bar, showing the route node in full screen mode. This means that all information can be removed from the interface except for the image, removing all distractions from the most important aspect of the route node display. Full screen mode also maximises the amount of the screen used by useful information. The improved route node display is shown in figure 10.3.
10.4 Improving tracking system

Currently the tracking system displays the closest route node to the current location. This is undesirable. A new tracking mode has been created in addition to the current method. Both modes have been left in the system so that the research group will be able to carry out extensive tests of the two systems running side by side. A “Tracking Mode” setting will be used to decide which tracking mode is used.

10.4.1 Tracking modes

It has been requested that a new tracking mode be added into Daisy. Rather than removing the old tracking system a setting has been added to allow the user to choose between either the original tracking mode or the new tracking mode. It will be possible to switch between the two tracking modes using the settings interface in the administrator system.

Tracking mode 1

This is the existing tracking mode where as the location changes the route node displayed also changes. This mode uses the current location and locations of route nodes to calculate which is the closest route node. The closest route node to the current location is displayed on the route node display.
Tracking mode 2

Following the user meeting a second tracking mode was suggested. This mode is used to wait for the user to be aware that they have reached a particular point in the route before the next instruction and route node display is presented to them. The tracking system is used to indicate to the user that they are travelling in the correct direction and then notify them when they arrive at the each route point.

Another proximity setting has been added to the setting store. This proximity value is used as a threshold to test if the reported location is considered to be at a particular route node or not. If the user is considered to be at the route node the display background is changed to blue. Through this display change and calling the notification function (see section 10.6) the user is informed that they have arrived at the required position. After this notification the user can then move to the next instruction. An example route node display tracked to the correct location is shown in figure 10.4.

![Figure 10.4: Route node display at required route node](image1)

To inform the user if they are moving closer to or further away from their destination either red or green bars are displayed at the top and the bottom of the screen. An example route node display during tracking is shown in figure 10.5.

![Figure 10.5: Route node display when tracking](image2)

The work-flow associated with this tracking mode can be seen in figure ??.. This diagram illustrates to order in which components are displayed on the canvas, assuming that the canvas is initially clear.

This alternative tracking mode will give the research group a better idea of
how Daisy can be used in guiding the users. As this is a prototype application in an experimental project both tracking systems have been included so that a more informed decision can be made about the potential uses of Daisy.

10.4.2 GPS signal fluctuations

As well as improving the tracking mode, the way in which location changes are detected has also been improved. Rather than changing the reported location as soon as a GPS signal changes the tracker checks that the location change is maintained over a number of consecutive location reports from the GPS device.

A “Location Certainty” setting has been used to specify the number of times that a changed location is reported before it is accepted. The default value has been set at 3 reports of a change of location before it is accepted. Like other settings this can still be modified by administrators.

By using the change certainty in this way random fluctuations in the signal from the GPS device do not affect the user. They go unnoticed as the changed location is never reported to the rest of the Daisy system using the tracker listeners.

10.5 Assistance feature

An assistance interface has been provided for the Daisy system. This interface can be loaded from any point in the user system. The assistance interface gives the user the option of sending for assistance or returning to the previous page without requesting assistance.

Assistance is requested using a text message. The ability to send a text message is provided by the package `javax.wireless.messaging` which is a part of the Wireless Messaging API. A message is constructed and then sent to
a specified number.

The message sent is comprised of:

• The name of the user needing assistance (taken from settings).
• The location of the user if available (taken from tracker).

After the message has been sent the user is returned to the interface in use before assistance was requested. It is also possible to return from this screen without seeking assistance.

![Assistance interface](image)

Figure 10.7: Assistance interface

### 10.6 Notification of change of location

Notifications are needed to be given to the user when the location changes in tracking mode 1 or the user arrives at the intended location in tracking mode 2 (see section 10.4.1 for details on tracking modes. The notification is required to draw the user’s attention to the phone for a number of reasons:

• A user exposing their phone all the time becomes vulnerable to theft. It would be better if they could keep the phone in a pocket or around their neck rather than holding it out in front of them at all times.
• If looking at the phone at all times the user becomes unaware of their surroundings. This may put them in danger or cause them to miss important landmarks that they should be looking out for.
• The user may become distracted and require some stimulus to draw their attention back to Daisy.

There are three methods of notification that have been considered. Each one stimulates a different sense to the user.

**Audio tone**: the phone can play some audio which will attract the users attention if they are not looking at the screen. A number of different forms of audio are available such as ring-tones, audio samples or more simple beeps and tones. A beep has been chosen as it is very simple and can be played very easily to quickly get the users attention.
**Vibration:** many phones have vibration functions. This is an ideal way of drawing the users attention to the phone as it does not require the user to see the phone, meaning it can be easily concealed, or hear the phone meaning it is still effective in noisy environments. Unfortunately the Nokia 6600 does not support vibration. APIs are available but they have no effect as the functionality is not supported by the version of Java supported. If Daisy is deployed on further devices this operation will work and a setting has been provided in the settings store so that the duration of vibration can be specified. The problem with the vibrate function is one of the known issues on the Nokia 6600 (issue 2.9)\(^\text{[15]}\).

**Flashing screen back-light:** if the user has the phone in their line of sight but are not necessarily looking straight at it screen flashing the screen could draw their attention. Unfortunately again, this feature does not work correctly on the Nokia 6600 and always results in the screen back-light turning off after flashing. Once off it becomes very difficult to re-activate it, only coming back if the application loses focus. This is an undesirable effect as the user can then not see what is on the screen so the flashing back-light feature has not been used.

### 10.7 Back-light always on

Ideally whenever Daisy is in use the back-light should be left illuminated. This allows the users to better see the content that is displayed on the screen. Under normal operation the back-light turns itself off to save battery life.

The back-light can be manipulated using the Nokia UI API method:

```java
setLights(int num, int level)
```

In the Nokia 6600 known issues document\(^\text{[15]}\) it is recommended that this package should not be used. Because of this the back-light always on feature has been omitted.

Utilities exist for mobile phones that leave the back-light permanently on. Some phones even have built in settings to control the back-light. With current technical resources this problem cannot be solved from within Daisy so it is recommended that an external approach be used if the feature is still seen to be desirable.

### 10.8 Using intuitive input

In the first prototype input into the Daisy system was mostly through command menus activated by the left and right soft-keys. It is desired that input be simplified wherever possible, ideally making input intuitive to the users.

User input from the keypad is handled by the different kinds of interface in different ways. **Form** interfaces leave little scope for handling key presses from the user. This does not pose a problem as having changed the route node display to a canvas, forms are no longer used in the user system. **List** interfaces allow a command to be associated with the “select” action, activated when an item from
the list is selected by pressing down on the joystick. **Canvas** interfaces allow handling of any key press event by overriding the `keyPressed(int keyCode)` method declared in **Canvas**. The `keyCode` parameter holds a constant to represent the key that was pressed.

Intuitive input has been added to the system by associated commands with keys that you would expect to perform such an action. For example, a selection command can be associated with the “select” action on the joystick. Also, when navigating through nodes “Next” and “Previous” actions can be associated with the directional commands on the joystick making movement through a list of route nodes feel more natural to a user than selecting the commands from a menu.

Intuitive input has been attempted in Lists using:

- **Route selection interface:**
  - “Select” command selects the currently highlighted route from the list of routes.
- **Route list interface:**
  - Select command opens the currently highlighted node from the list of route nodes.

More key presses have been handled in Canvases with the `keyPressed(int keyCode)` method.

- **Daisy user system mains screen:**
  - “Select” shows the selected route if one is loaded or brings up the select route interface if not.
- **Route node display**
  - “Select” plays the instruction media again.
  - “Left” goes back to the previous route node.
  - “Right” goes forwards to the next route node.
  - “Down” displays the description of the node.
  - “Up” toggles between normal and full screen mode.
  - “0” goes to the assistance interface.
- **Assistance interface**
  - “Select” sends the assistance message.
Chapter 11

Daisy Administrator System

The Daisy administrator system is designed so that helpers to the Daisy users can set up the user system for them. This includes specifying settings and loading in routes and media. The admin system is built of a number of utilities and then tools for creating routes and media.

Figure 11.1: Administrator system main screen

11.1 Utilities

As already mentioned in the “Architectural Issues” (section 7) of this report a number of utilities are included with the administrator system that manage the core functionality behind the Daisy system. The utilities include:

- Store management tool (section 7.1)
- Settings modification tool (section 7.2)
- GPS status display (section 7.5.1)
- Log view utility (section 7.6)
11.2 Route capture

Route capture can be used to either create entirely new routes or modify existing routes. Routes can be modified to add new route nodes before or after any existing node, delete a route node from a route or to modify any existing route node. New routes are created by providing an empty route and then allowing the administrator to add in new routes nodes.

The route node is changed using a route node modification form. When an existing route node is modified it is possible to change:

- Title of the route node using a text field.
- Description of the route node using a text field.
- Instruction or Orientation media by selecting a new media item from the media library (see section 11.3).
- Change the location of the node by connecting to a GPS device to obtain a new location.

Having made changes to the route node it is possible to either store them or just return to the route without saving the changes to a particular node. If changes are to be saved the modifications are sent back to the route using a RouteListener. This interface is implemented by classes wishing to receive new route node details. When the route node has been modified the changed node is sent back to the listener. If no changes are required then nothing is sent back to the listener. When the listener receives a route node it updates the route appropriately by either inserting a node or updating an existing node.

When a route is created it is stored in the persistent store on the device. Any changes to the route are immediately updated in the persistent store so can then be seen in the user system.

11.3 Media Library

The media library is another utility provided in the Daisy administrator system. It is used to show all of the media present in the persistent store. The media is initially displayed in a List user interface which includes a thumbnail image for each piece of media and the name of that media. For audio and video media an image has been provided to give a visual representation of the type of media as otherwise no thumbnail would have been available. The media library display can be seen in figure 11.2.

The media library user interface offers a number of operations to manage the media stored on the device:

Select Media: this is used when the media library is being used to select an item of media to be used in a route node. When the select option is enabled and activated it uses a reference to a MediaListener object to return the name of the selected media item. Any class wishing to receive such a piece of media must implement the MediaListener interface.
Display Media: any item of media can be displayed from the library. A new display opens to show the selected piece of media. Depending on the type of media a different display is required. The three types of display available are:

- Image displays which show the image represented by the image media.
- Audio displays that play the audio using the media player provided with the media while showing the default audio image.
- Video displays that play the video media using the media player provided with the video media. No default image is required for this display as the video provides visual information unlike audio media.

Capture Media: it is possible to capture image and audio media using Daisy. Details of media capture are given in section 11.4.

Rename media: allows the user to rename the currently selected piece of media from the library. This does not update the names of media in routes, it is intended for renaming newly captured media.

Refresh: reloads the media library from the persistent store.

Store current: clears the persistent store and rewrites the current media library to the store.

Delete media: removes the selected media item from the library (the media is removed from both the display and the persistent store).

Empty library: empties the entire media library removing all items from the persistent store.

11.4 Media Capture

Although there are three kinds of media available for display, as many phones including the Nokia 6600, do not yet support video capture only photo and audio capture have been provided with the admin system.

Media capture is performed by creating a new thread to perform the capture. This thread is needed as otherwise the capture would take complete control of...
the phone and not allow the user to control the capture. The user would not be able to perform actions such as capturing a photograph or stopping a recording without the capture performed in a separate thread of execution.

The thread captures the media data and returns a block of byte data to represent the media. This is put into an appropriate Media object of the correct subtype, either ImageMedia or AudioMedia. When media is captured it is given the default name “New Media”.

### 11.4.1 Image Capture

Image capture is performed using a snapshot from a VideoControl. Even though video capture is not supported the video display can be used from the camera to show what the camera is currently being pointed at. The snapshot takes a raw image from whatever the camera is currently pointed at. An example of how image capture is performed is given in listing 11.1.

It is possible to set the camera to take images of different sizes and formats. This is done by providing a different string to the method to specify the image type. The parameters that can be modified include encoding, height and width. Examples of encodings are given in listing 11.2.

Unfortunately it is not possible to use the camera at its best possible setting as this is not supported. The known issues document for the Nokia 6600 states the following:
Listing 11.2: Snapshot formats

```java
byte[] raw = null;

// Use the default encoding (png, maximum size)
raw = vc.getSnapshot(null);

// Use the default encoding (png) 80 x 60 pixels
raw = vc.getSnapshot("width=160&height=120");

// Jpeg encoding, image size 120 x 90 pixels
vc_.getSnapshot("encoding=jpeg&width=120&height=90");

// Bitmap encoding, maximum size
vc_.getSnapshot("encoding=bmp");
```

“...The maximum size that can be captured depends on the free heap memory available...”

Unfortunately due to limited space in the heap each time the camera has been used at its full 640x480 resolution it has returned an out of memory exception. This is not a great setback as the screen size is only 178x208 so would not be able to display such images anyway. It has been found that the largest image size that can be capture safely without risk of an exception is the default size of 160x120 pixels. As this is almost the complete width of the screen it is considered to be acceptable. There is no advantage in increasing the height of the image captured as this would merely cause the image to be stretched which would distort the image and reduce the quality.

Various studies into the performance of the Nokia 6600 when capturing images have been conducted. One such study was by the Embedded Interaction Research Group [19]. There study showed that the best quality image is encoded using the PNG format. Even though this format produces the largest files and memory is at a premium it is important that high quality images are presented to the users. To this end PNG images will be captured and displayed as quality of image is more important in this application than the memory used to store them.

11.4.2 Audio Capture

When capturing audio it is important to consider the format in which the audio data will be encoded. There are a number of different encodings available including ‘WAV’ and ‘AMR’. Most phones in production support all three of these encodings however this is another feature that must be checked for compatibility when deploying to new devices.

The default audio encoding is ‘WAV’ using Pulse Code Modulation (PCM).
A problem associated with this kind of audio capture in J2ME has been named the “Mickey Mouse effect”. This is when the audio recorded does not accurately reflect the original source, instead it sounds like it has been sped up resulting in high pitch tones making people sound like Mickey Mouse. The problem comes from the sampling rate used in the capture. The best setting to use to avoid the problem is a sample rate of 8000Hz. The details of this problem can be found the Known Issues document for the Nokia 6600 (Issue 3.2)[15].

Having solved the problem with sampling there is still an issue with the size of the data captured even for small media samples. With the ‘WAV’ audio produced using 8000Hz samples at 16bits per sample, one second of audio will use 128,000 bits. A decent length sample will quickly use a lot of memory which may cause a big problem due to our lack of storage space.

A better encoding available on the phone is the Adaptive Multi-Rate (AMR) encoding. This stores audio data in a much more memory efficient manner making it far more suitable for our purposes. The only issue with this codec is that it is not commonly used on desktop computers. This means that any samples recorded outside the Daisy system will have to be converted to AMR using an appropriate tool before being used as Daisy media.

‘AMR’ was chosen as the encoding to use as the smaller media files produced offer a far greater advantage than any problems with using the less common encoding cause. Audio samples are taken using this encoding using a Player as in listing 11.3.

The output stream used in this case is a ByteArrayOutputStream. This stream simply places the recorded data into a growing byte array. At the end of capture the byte array can be taken from the stream to be stored as media.

Listing 11.3: Audio Capture

```java
// Create a player to use in capture
Player p =
    Manager.createPlayer("capture://audio?encoding=amr");
p.realize();

// Get a control and set the output stream for data
RecordControl rc
    = (RecordControl) p.getControl("RecordControl");
rc.setOutputStream(os);

// Start recording
rc.startRecord();
p.start();
```

Controlling Media Capture

Media capture is launched from the “Modify Route Node” GUI to either capture Orientation or Instruction media. The media capture is controlled by an “Image Capture” GUI and “Audio Capture” GUI (figure 11.3).
Once the media has been captured the byte data is put into a Daisy Media item which is stored in the media library. The media item is identified by a unique name in the media library. With newly captured media this name is of the form “New Media(x)”, where x is the next free media number. Media names can be changed in the library after the media has been captured.

The media capture GUIs are each initialised with an instance of a MediaListener that is a reference to the object that wants the media after it has been captured. When the media capture GUI is dismissed after capture has been performed a call is made to the setMedia(String) method of the MediaListener that tells the listener the name of the new media item in the library.

11.5 Transferring Route and Media Data

It is desired that the Daisy application should be able to display a number of different routes. This requires several different sets of route and media data. As there is only limited space available to store the routes on the phone it is necessary to be able to move routes onto and off of the phone.

The method used to solve this problem initially involved accessing data on a server using HTTP transferring all required data onto the phone. This would prove to be costly when transferring large amounts of data over the mobile network and slow as the rate at which data can be copied using this method is limited to 32-48Kbps for GPRS networks used.

A better solution is to use the Bluetooth capability provided by most mobile phones. This will allow data to be transferred between co-located Bluetooth devices (including both mobile phones and desktop computers). This method has advantages of being free, using radio links directly between devices, and much faster than GPRS with transfer speeds of between 64kbps and 1Mbps. A disadvantage of this method is that you need to be within 10m of the device (100m for some more powerful devices) that you are communicating with. This restriction means Bluetooth transfer of route and media data is not possible when Daisy is in use “out and about”. To solve this problem HTTP transfer has been left as apart of the system although it can easily be disabled from the settings menus.
11.5.1 Bluetooth architecture

To transfer data between devices they need to form client/server pairs. The server creates a serial port service and then waits for a client to connect to this service. The client searches for a device offering such a service and opens a serial port connection using a Bluetooth link to the server.

The service is identified using a universally unique identifier. This means that when the service record is created its identifier prevents the Daisy transfer service being confused with other services offered. The identifier can be created using the unix command `uuidgen -t`. This creates an identifier of the form: `43824f34-8cbf-11d9-85c7-00304823beab`. This is the identifier used with the Daisy transfer service.

To discover a transfer server you first need to perform a device discovery for local devices. Details of device discovery are included in section 7.2.1. Once a device has been discovered the system must check that the correct service record exist on this device. This is done in a similar way to searching for devices. The same DiscoveryListener interface is implemented and a discovery agent is used to search a particular device for services. All of the services found on a device are reported back to the class that implemented the DiscoveryListener. If the Daisy service record is returned then the device can be used to transfer Daisy data.

11.5.2 Docking Stations

Data transfer is managed by a docking service. A docking service can either be a docking server or a docking client as shown in figure 11.4.

![Figure 11.4: Daisy Docking Classes](image)

Docking server

The docking server first creates the service record and registers this for discovery with the local device. The server then waits for a connection to be opened, blocking until this is received. So that the user can still use the Daisy system while the server is blocked the docking server is run as a thread. This means that the user interface can still be used to give input to Daisy allowing the server to be stopped if required.
When a client opens a connection with the server, a new connection is established. The \texttt{DataTransfer} class is then provided with the connection and type of data to be sent so that it can send the data on behalf of the docking station. This is done using a \texttt{send(int, Connection)} method call to a \texttt{DataTransfer} object. The data type to be sent is specified using an integer constant. Values of the constant include \texttt{MEDIA\_DATA}, \texttt{ROUTE\_DATA} and \texttt{ALL\_DATA}.

After data has been sent using one connection the server then goes back to waiting for new connections to be established. This allows data to be sent to multiple clients without the need to restart the server.

\textbf{Docking client}

Before starting a docking client the service record must be set in the \texttt{DockingClient} object. The service record is found as described above then this is set using a method call to the docking client. When the docking client is started it uses the service record to open a serial port connection to the remote server that has already blocked, waiting for a connection. Once established the connection is passed to another \texttt{DataTransfer} object, this time using a \texttt{receive(Connection)} method call.

\textbf{Data Transfer: Send}

Using the connection passed to the \texttt{DataTransfer} object an output stream is created. The output stream is used to send byte data from the server to the client over the Bluetooth serial port connection. Having established the stream, the data required to be sent is converted into byte data using object serialisation. The data is now ready to be sent.

The data sent on the stream and the order in which it is sent is:

1. Type of data being sent (integer constant)
2. Length of byte data to be sent of a given type
3. Byte data of a given type

The length and byte data in steps 2 and 3 are sent repeatedly until all types of data have been sent. Once all data has been sent the connection can be closed.

\textbf{Data Transfer: Receive}

At the client the \texttt{DataTransfer} object uses the connection object passed to it to create an input stream to read data from the Bluetooth connection.

When the receiver starts it does not know what type of data it is going to receive. The first item received on the stream tells it what data to expect. Following the type value each chunk of byte data is received on the stream. Once all data has been received on the stream the byte data is resurrected to get copies to the original objects sent. The byte data is resurrected into objects based on the type variable that it received. The objects are then put into the
appropriate persistent store after ensuring that no name clashes in route or media occur.

Once all data has been received the Bluetooth connection to the server is closed so that the server can accept more connections and the client can return.

11.5.3 Problems with javax.bluetooth and Nokia 6600

There are two known issues with the Nokia 6600 and Bluetooth serial ports. These had to be overcome in order to produce a working docking station utility. The known issues are given in more detail in issue 2.30 “Using Bluetooth Serial Port in MIDlets” in the known issues document for the Nokia 6600[15].

Closing input streams

When using the read() method on serial port input streams it has been found that IOExceptions may be unexpectedly thrown. This is caused by the sending end of the connection being closed before the receiver has finished reading from it.

In order to solve this problem is necessary that connections are not closed at the server end before all data has been received at the receiver.

Reading input streams

A second issue is that only 512 bytes can be successfully read at a time from an input stream. To solve this problem the data has been read in chunks of 512 bytes until the entire array has been read.

Generic Solution

A generic solution to problems with Bluetooth serial ports is to use the L2CAP (Logical Link Controller and Adaption Protocol). This is a higher level protocol than RFCOMM used for serial ports providing quality of service information and controlling packet segmentation and reassembly. In short it enables provides higher quality data services between Bluetooth devices.

The reason this protocol has not been used is that it is desired that data should be sent between a mobile device and a desktop machine. This involves communicating between J2ME (Java 2 Micro Edition) on the mobile devices and J2SE (Java 2 Standard Edition) on a desktop machine. It is only possible to perform this communication using RFCOMM as there is currently very limited support for L2CAP on J2SE.

11.5.4 J2SE, J2ME and Bluetooth Transfer

To use Bluetooth serial ports the Java javax.bluetooth package is needed. This package is defined in the API specification JSR-82[20]. This specification was defined for use in J2ME. The package has not been implemented for J2SE. In order to connect a desktop PC to a mobile device the same package must be made available in J2SE on the desktop machine.
Listing 11.4: Docking console server output

Loading Media Library... done.

Please select a command from:
1 = Send data to mobile device
2 = Receive data from mobile device
Command: 1

Send data to mobile device.
Docking station server URL:
btsp://localhost:<<uuid>>;name=DaisyDockingTransfer

[WARNING] You are using BlueCove Connector [WARNING]

Waiting for client connection... accepted a client connection.

Sending data... data sent.
Close current connection.

Waiting for client connection...

A number of commercial and free implementations of the `javax.bluetooth` package exist for J2SE. I chose to use the BlueCove implementation. This is an implementation of the Bluetooth API for Java using the Windows XP SP2 Bluetooth stack. Although this severely limits the hardware that can be used with the system, as it must be compatible with the Windows Bluetooth stack, it will serve as a test platform to prove that the concept works. Future implementations would be able to use other implementations of the Bluetooth stack that are compatible with more hardware.

As the package is now provided in both J2ME and J2SE the applications running on the mobile device and desktop machine can share the same classes when transferring code. In each transfer operation one will work as a server and the other as a client.

The only changes that had to be made in moving the data transfer utility onto a desktop machine were in storing the data and the interface used to display the data. Instead of records being stored in a persistent store they are now stored in files on the local disk. The interface on the desktop machine is a simple console. The console can be seen in for the docking server in listing 11.4 and the docking client in listing 11.5.

Input is all very simple and from the command line. A more complex interface has not been developed as it currently seen to be unnecessary at this stage.

Listing 11.5: Docking console client output

Loading Media Library... done.

Please select a command from:
1 = Send data to mobile device
2 = Receive data from mobile device
Command: 2

Receive data from mobile device. Searching for Bluetooth devices.
0: Nokia 3230
1: Nokia 6600 RSS
2:
Enquiry completed, 3 devices were found.

Please select a device (0–2): 1

Searching for Bluetooth services.

Searching for services on: Nokia 6600 RSS Service found:
btspp://006057d97190:4;authenticate=false;encrypt=false Service found:
btspp://006057d97190:4;authenticate=false;encrypt=false
Service search completed

Server URL: btspp://006057d97190:4;authenticate=true;encrypt=false

[WARNING] You are using BlueCove Connector [WARNING]

Receiving data... 224910 bytes expected at receiver.
Data received.
Close current connection.
Media Records from Desktop PC

Utilities have been provided to construct and extract Daisy media records. Media can be extracted from media records sent to the desktop machine. This can be used to retrieve media files captured using the administrator system. More usefully, media records can be created to be deployed to a mobile device using media files on the desktop PC. They are correctly formatted with the name of the media, the media type and the byte data so that they can be sent via Bluetooth for use in routes.
Part III

Evaluation
Chapter 12

Evaluation

This evaluation looks at the Daisy system created. It comments on the features that work as well as those that don’t while looking back at the specification to see what was expected. This is a mainly qualitative analysis however some quantitative results from the performance of the Daisy system have been added to emphasis certain points. Additional evaluation of the system was carried out after the meeting with member of the research group as seen in section 9.

12.1 Specified vs. Delivered Functionality

The requirements for the project were set out in the specification (chapter 3). Looking back after implementation there have been a number of changes between what was requested and what has been delivered. Reasons for these changes include technical difficulties forcing design decisions to limit the scope of the project and also some changes in the user requirements based on the meeting held with the users.

The original specification was divided into milestones. By considering each milestone it will be possible to see what is available from Daisy and what has been omitted. The successes and failures of each aspect are also considered here. In addition to descriptions of delivered functionality, table 12.1 summarises the results.

12.1.1 Milestone 1

This milestone required the core Daisy user system to be developed. The main aspects required in the application were specified as:

**GPS connection and control system:**

GPS connection has been provided using the trackers (see section 7.5.2). Daisy is capable of connecting to a GPS device and using a parser it can extract locations from the stream of data from the device.

To test the tracking provided by Daisy a path around The Queens Tennis Club in West London was followed. Periodically the longitude and latitude of a
<table>
<thead>
<tr>
<th>Milestone 1</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS connection and control system</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>Pre-trip planning and In-trip navigation</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>User customisation</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Milestone 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile administrator system</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Milestone 3</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide video media</td>
<td>PARTLY COMPLETED</td>
</tr>
<tr>
<td>Multiple pieces of media for a route node</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>Assistance feature</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Milestone 4</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop administrator system</td>
<td>PARTLY COMPLETED</td>
</tr>
</tbody>
</table>

Table 12.1: Results of standard use tests for the Daisy user system

The position was recorded from Daisy to ensure that the tracking recorded correct location values. The results can be seen in appendix A.

It was found that Daisy accurately recorded the position of each of the sample points along the route. All of the locations reported identified the correct location of where the data had been collected. This small sample gives an indication of the accuracy of the tracking system. It will allow reporting of locations down to a few meters so that routes can be specified precisely.

A limitation of the tracking system occurs when the GPS device is first switched on. As you would expect the GPS cannot get a fix on its location immediately. In fact it has been found to take some time before the device does get an accurate value for its current location. According to the specification of the Fortuna Clip-On Bluetooth GPS used in this project the times for acquisition of location co-ordinates are:

<table>
<thead>
<tr>
<th>Start State</th>
<th>Acquisition Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reacquisition</td>
<td>0.1 sec</td>
</tr>
<tr>
<td>Cold start</td>
<td>45 sec</td>
</tr>
<tr>
<td>Warm start</td>
<td>38 sec</td>
</tr>
<tr>
<td>Hot start</td>
<td>8 sec</td>
</tr>
</tbody>
</table>

Table 12.2: Acquisition Times of Fortuna GPS

The acquisition times in Table 12.2 are given that the device is stationary under open sky so are best case averages. In reality, with interference from surrounding buildings and trees acquisition can take much longer than stated here.

This problem means that the users would not receive location reports for some time after they connect to the GPS device. If the users are waiting for this input and do not receive it they may become agitated and lose faith in the system. Alternatively they may assume that their location has been set and use the system as if it had. In this case they will become disorientated if the
route does not track them correctly as they attempt to follow it.

A similar problem occurs when signal is lost from the GPS. This may occur if the users pass under a covered or extremely built up area and satellite signals are temporarily lost. Again in this situation it would be desirable to inform the user they are no longer being tracked using some form of visual indication.

As this is a problem with the GPS device rather than the Daisy system it is necessary to cope with the problem rather than to solve it. A notification of when the GPS is connected is suggested. This could be something like adding an icon to the screen when connected that is easily associated with tracking.

Other applications using GPS generally report when there is poor satellite reception which will cause problems in pin pointing the current location. Although the tracker reports this information it is not displayed to the user as it was thought to be too complicated and would only confuse them. This is still a valid point as you do not want to overload the users with information. This issue should be discussed with the accessibility research group as it concerns communication with the users.

**Pre-trip planning and In-trip navigation**

Pre-trip planning is provided by using the interfaces without tracking connected. It is possible to view the route one node at a time, manually moving through the route.

In-trip navigation is provided when the GPS is connected. The tracker allows you to follow routes, moving between the nodes on the route. Reassurance is given to the user by the system indicating whether they are moving in the correct direction and then when they arrive at route points.

A limitation of the in-trip tracking system is that Daisy gives no assistance when the user strays considerably of their intended route. There is no method of directing the user back to the route that they should be following. This is likely to cause the users problems as they will have no assistance should they deviate from the route which may cause them to panic making their situation worse.

One of the successes of this project has been in Daisy's ability to show different forms of media and accurately describe routes. Audio and video media can be played to the user giving the user much more information than if only images were available.

**User customisation**

A settings configuration tool has been provided with Daisy. This allows administrators to adjust the set up of the system to improve its performance. The settings available range from the certainty used when collecting location information and the distance that you use to decide when a user has arrive at a particular route point to specifying destination phone numbers for assistance messages. All settings are fully modifiable.
12.1.2 Milestone 2

Mobile administrator system

The administrator system provides methods of media (audio and image) capture as well as route specification. As required this is a portable piece of software written for the Nokia 6600 in particular. This will allow travel trainers to go out and acquire routes that the Daisy users will subsequently be able to follow.

Both media and routes are stored onto the phone and are immediately ready to be used by the Daisy user system. The tracker is used to obtain locations for route nodes. Route nodes can be inserted at any point in an existing or new route or deleted from the route. This gives administrators the flexibility so they can specify their routes as they wish.

12.1.3 Milestone 3

Provide video media

As video capture is not supported by Daisy due to limitations with the Java implementation on the devices used, extensive video playback options have not been provided. Although it is possible to play a video in Daisy, little work has been put into this area and the wide variety of controls originally intended to be present (Pause, Stop, Rewind, Forward) have not been implemented.

Only a simple player has been provided for video. This player is very similar to that provided for audio media except that the visual display has been added. When it was discovered that video capture was not possible it was decided that video options where no longer an important concern as video will rarely be used in Daisy. The basic video player has been provided so that in the future, should video capture become available the additional video control functionality can be added relatively easily.

It is disappointing that video media cannot be used as the additional media type would have made it far easier to communicate with the users. The idea behind Daisy is that information should be able to be presented to users in the most appropriate way possible. Removing the choice of video media takes away from the overall usefulness of the system. Fortunately image and audio media are fully supported and can be used as alternatives to video media providing both visual and audio stimulus to the user as an alternative to video.

Multiple pieces of media for a route node

Originally it was intended that Daisy should allow multiple pieces of media to be supplied for a route node. The problem with this approach is in how the media would be presented to the user. Decisions would have been required about the order in which media should be displayed based on some measure of importance. The method of changing between media items could use methods such as changing after a given length of time or after some user input. The most important factor would be to avoid confusing the user; this may be very difficult as many different pieces of media will all trying to be get across the same intention using different means and content. The users may not be able
to take in all of this information in which case they would become confused and lose confidence in the Daisy system.

An alternative method of associating media with routes was developed following the user meeting. This involved having two different kinds of media for each route node, a piece of instruction media (audio) and orientation media (image). This approach gives the two forms of information about the users location and which direction they should follow. As the media comes in two different forms (audio and image) the user will be less likely to get confused as the media items are distinct from one another. By providing both visual and auditory stimulus the user gathers more information with less chance of confusion.

A limitation of only having orientation and instruction media rather than any number of pieces of media is that the user is presented with fewer pieces of information find their way. If they cannot locate themselves based on the 2 media items there are no alternatives to use. Multiple pieces of media would have given them more options increasing their chances of recognising their surroundings. It is important to balance the number of pieces of information given to help the user and the number of pieces of media that would begin to confuse them. Using instruction and orientation media for each node is seen to provide sufficient information, provided it is chosen carefully, without overloading the user.

12.1.4 Assistance Feature

The assistance feature has been provided to send a text message from the Daisy system to a mobile phone number specified in the settings. The assistance sends details of the users location and so that a helper can establish how best to help when they contact the user.

A problem can be seen with the assistance feature when considering how it is activated. Currently assistance can only be requested from a menu in the user system or by pressing the “#” key whilst displaying a route. This may cause the users difficulty as this method of activation is not intuitive. It is likely that they will forget how to activate the assistance feature especially if they are already in a state of confusion. An automatic assistance feature may be better than requiring the user to activate it from the phones interface.

Possible reasons for automatically activating the assistance feature could include:

- Location recorded being a considerable distance from the route indicating the user has lost their way. A threshold value for determining if the user has left the route could be stored and modified in the Daisy settings.

- Lack of progression through a route such as a user remaining at a single route node or moving between a subset of route nodes repeatedly.

- Daisy system being given repeated or multiple commands such as selecting different routes or quickly moving around a single route as if the user did not know what they wanted to do. This would be used to detect user disorientation as their behaviour would indicate they are struggling to use the Daisy system.
12.1.5 Milestone 4

This milestone was concerned with producing a desktop application to manage routes captured by a mobile device. A limited set of the functionality specified has been implemented for this tool. It was seen that in the time provided it was important to specify the most important functionality. To this end a docking station has been created.

The docking station allows captured data to be sent to and from the mobile administrator system to be stored. The docking station can also send stored data back to the mobile system for it to be used in the Daisy user system. This is done using a Bluetooth connection so is free and fast. The alternative method of getting data to the phone is using HTTP which is extremely slow and requires a lot of data to be transferred across the mobile network so becomes expensive too.

As the administrator tool is only a docking station it provides no features to manipulate route data.

12.1.6 Deliverables

The evaluation criteria specified that a number of deliverables were required from the project. Along with this report the deliverables expected were:

- DAISY software for users to run on mobile device
- Mobile administrator software to acquire images
- PC based software to manage routes
- User guide for both DAISY and administrator software
- Feasibility report discussing the proposed extensions

What has actually been provided are the mobile user and administrator system and the PC based management system.

Only a user guide for the Daisy user system has been produced due to time constraints on the project. This can be seen in appendix C. The administrator system has not been documented as a lot of the features are similar to those in the user system. Otherwise menus and instructions on screen should be sufficient for travel trainers.

In regards to the feasibility report on the extensions to the project, a discussion of their potential is included later in the conclusion (section 13.1).

12.2 Evaluation of Daisy Operation

In order to evaluate the quality of the Daisy system provided a number of areas of its operation have been considered.
12.2.1 Standard Use Tests

A number of tests were specified in the evaluation criteria. These tests were intended to ensure that the core functionality required by Daisy (described in the milestones above) is present. The results of these tests are shown in tables 12.3 and 12.4.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct GPS location is established and used</td>
<td>✓</td>
</tr>
<tr>
<td>A route to a destination can be loaded into Daisy</td>
<td>✓</td>
</tr>
<tr>
<td>Routes can be stepped though as a sequence of directions</td>
<td>✓</td>
</tr>
<tr>
<td>At a given location the correct media is displayed</td>
<td>✓</td>
</tr>
<tr>
<td>Multiple items of media at a single route node can be viewed in turn</td>
<td>✓</td>
</tr>
<tr>
<td>Media changes when appropriate as the user moves</td>
<td>✓</td>
</tr>
<tr>
<td>When off the route no direction is given but a panic/help service is offered</td>
<td>χ</td>
</tr>
<tr>
<td>Media is viewed appropriately depending on format (image, audio, video)</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 12.3: Results of standard use tests for the Daisy user system

All but one of the standard use tests operate correctly. The only test that failed is in detecting when the user strays from a route and providing assistance. Details of this problem are given in section ??.

As the standard use cases mostly passed this tells us that a majority of the core functionality has been implemented.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photographs can be captured and stored</td>
<td>✓</td>
</tr>
<tr>
<td>Audio can be captured and stored</td>
<td>✓</td>
</tr>
<tr>
<td>Video can be captured and stored</td>
<td>χ</td>
</tr>
<tr>
<td>Locations stored with media are correct</td>
<td>✓</td>
</tr>
<tr>
<td>Routes can be created from data acquired by mobile system</td>
<td>✓</td>
</tr>
<tr>
<td>Order of nodes in a route can be rearranged</td>
<td>χ</td>
</tr>
<tr>
<td>Two routes can be merged</td>
<td>χ</td>
</tr>
<tr>
<td>Multiple routes can be stored and loaded when required</td>
<td>✓</td>
</tr>
<tr>
<td>Route and media data can be imported from an external store</td>
<td>✓</td>
</tr>
<tr>
<td>Route and media data can be exported to an external store</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 12.4: Results of standard use tests for the Daisy administrator system

Again most of these tests have been satisfied indicating the required functionality is present.

Firstly due to technical limitations video capture is not possible. This meant that although video can be played and stored it cannot be captured.

The second missing functions are in manipulating stored routes. It is not possible to manipulate the routes in the current administrator system. The only functions available are to insert a new node at any point in the route or to delete a route node. It is not possible to combine and rearrange routes. This functionality would be better on a desktop based administrator system where it would be easier to manipulate the routes using more complex input methods.
on the phone. Currently the desktop administrator software can only be used to send, receive and store route data rather than manipulating it.

12.2.2 Robustness and Stress Tests

In the evaluation criteria several stress tests were specified. These are intended to show whether or not the system stands up to extreme conditions. The results of the stress tests can be seen in table 12.5 a brief description of each are shown below.

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequently moving between locations on route</td>
<td>✓</td>
</tr>
<tr>
<td>Changing route node and interrupting media</td>
<td>χ</td>
</tr>
<tr>
<td>Moving on and off the known route</td>
<td>✓</td>
</tr>
<tr>
<td>Working in areas where no GPS signal is available</td>
<td>χ</td>
</tr>
<tr>
<td>Turning off the GPS device while tracking</td>
<td>χ</td>
</tr>
<tr>
<td>Closing the application while following a route</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 12.5: Results of stress tests (pass: ✓ or fail: χ)

**Frequently moving between locations on route**: the trackers correctly follows the users location. As the location reports come in the tracking system updates system correctly reporting the progress to the next intended route node. Overall this does not cause a problem to Daisy.

**Changing route node and interrupting media**: If media is being played (audio or video) when the route node displayed changed the media is stopped before the new node is displayed.

**Moving on and off the known route**: The tracking features only tell the user if they are moving away from the intended route nodes. The system will merely indicate that the user is far away from the route.

**Working in areas where no GPS signal is available**: When GPS signal is not available there is no way for the user to track themselves. Currently the user is not informed of this so they may just wait for the signal to become available which it never will.

**Turning of the GPS device while tracking**: The tracker will simply stop reporting locations in this situation. No location updates are given to the user but since they are not informed that the GPS device is no longer connected this test is considered to have failed.

**Closing the application while following a route**: When the application is closed and restarted the same route will be loaded however the position on the route will not be restored.

It can be seen above that Daisy does not satisfy all of the stress tests. Daisy is not as robust as it was originally intended to be. During development a trade off had to be made between adding sufficient features to show the accessibility
research group the wide variety of features that were possible and ensuring that every aspect of the system had been rigourously tested to ensure robustness. As this is a prototype system the emphasis was put onto ensuring that the application gives all of the operation required by the users.

This is a failing of the system as to be suitable for the intended users Daisy needs to be able to stand up to these and more stress tests. It is important to remember that if the Daisy system fails the user has no support so will become disoriented.

The most important problem that must be handled are problems with the GPS. There must be someway to communicate with the user that tracking has failed or connection to the GPS has been lost. If this is not provided the user will still believe that they are being tracked and struggle when the system never indicates that they have arrived at their destination.

12.2.3 Performance

Little performance consideration has been made in the design of Daisy. Most operations are not computationally expensive and it has been found that once in operation Daisy is remains responsive, performing all functions almost instantaneously.

12.2.4 Tracking System

Locations are reported from the GPS device around once every second depending on the signals receive from satellites. As a change certainty value is used the location is not immediately sent to the tracker listeners. Before a location update is sent a location change has to be registered a number of times specified by the change certainty setting. It can be assumed that the change certainty is the time taken for a change in location to be registered.

For example, this manifests itself if you are in a particular location and then begin moving. Once moving the system will only recognise your change of position after the number a seconds equal to the change certainty. Subsequent location updates will continue to be reported at this interval.

As the Daisy system is designed for pedestrians on foot this is a suitable amount of time. In 3 seconds they do not move a great distance so even though the system may be a few steps behind you when walking fast uses will slow down and stop when they reach their destination allowing the system to catch up.

12.2.5 Persistent Store

The only area where the Daisy user system performance degrades is when loading in the data required from the persistent store. It takes considerable time to load routes and media associated with them, especially for longer routes with more route nodes requiring more data to be fetched from the persistent store.

Using time stamps associated with log entries it is possible to measure how long it takes to load routes. The results of loading several routes are shown in table 12.6.
Table 12.6: Times taken to load routes with different number of route nodes

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Time taken to load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 node</td>
<td>4 seconds</td>
</tr>
<tr>
<td>2 nodes</td>
<td>6 seconds</td>
</tr>
<tr>
<td>3 nodes</td>
<td>8 seconds</td>
</tr>
<tr>
<td>4 nodes</td>
<td>11 seconds</td>
</tr>
<tr>
<td>5 nodes</td>
<td>12 seconds</td>
</tr>
</tbody>
</table>

This time delay is not acceptable for the intended Daisy users as they will assume that the commands they issued did not work and start pressing other buttons on the phone to try and make something happen. This is likely to then activate others of the phones features, moving Daisy into a different state than they required or activating other phone applications on top of Daisy.

In order to combat this problem routes should be pre-loaded so that when they are required they can be loaded quickly. This could be done using an additional thread that predicts what routes and media are required so that they have been loaded into memory when they are required. It is not possible to fetch too much data into memory in advance as there is limited memory available.

**Handling Delays**

When delays such as these occur due to background processing going on some indication that the system is doing something should be given. A “Please Wait” message should be given that blocks any other user input until the operation has completed. This interface would let the user know that there command has been accepted and is being processed. Currently the system does not respond after certain operations due to the processing required. In these situations the user may attempt other operations rather than waiting. This problem is solved easily but is very important to make the system for user friendly.

**12.3 Usability of the System**

To assess the Daisy system the usability has been considered. As discussed in the background (section 2.3.1) it is important to get the human computer interaction right in order to make the system as suitable as possible for the users. Usability is considered for the Daisy user system as this is the part that will be used by those with cognitive difficulties. The administrator system will be used by travel trainers so it is less important that it is fully accessible.

The interfaces used in the Daisy system are shown in figure 12.1. The arrows indicate the possible flows between interfaces. Where the assistance interface has multiple arrows entering and leaving, the path followed is to always leave on the arrow to the interface that it was entered from.
12.3.1 Interface Standardisation

The users will find the system to be easier to use if they are presented with recognisable interfaces. If the interface style varies greatly throughout the system they will have difficulties interpreting the information presented to them on each one.

The standard user interface designs provided by java mean that all interfaces have a similar style. In order to further the recognition of Daisy an icon has been provided on each interface. This daisy icon makes any interface instantly recognisable as being a part of the Daisy system.

One way that interface standardisation may have been taken too far is that it is very difficult to distinguish between a list of routes on the route selection node and a list of nodes inside a route on the route list interface. It may be appropriate to remove the route list interface and only display routes a node at a time. This will not affect functionality but make the system a lot clearer for the Daisy users.

Figure 12.1: Daisy User System Interface Structure
12.3.2 User Input

As Daisy is executed on a mobile phone user input methods are very limited. The keys available are listed in the technical details chapter of this report (section 6.1). In addition to using keys for input, menus containing commands are accessible using the left-soft key.

Wherever possible keys have been used to provide shortcuts to menu items. This has two aims, firstly to make input into the system more intuitive that selecting commands from a list, and secondly to reduced the number of key presses required by users to activate different features. The keys assigned to commands are given in section 10.8.

When choosing which keys to use to control Daisy it is important to bear in mind what you would intuitively expect the key to do in that situation. If keys are given function similar to what you would expect that key to do in a given situation it becomes a lot easier to remember the key that your require. For example, when using lists the select button (pressing down on the joystick) should select the highlighted item from the list. Also when navigated route nodes left and right go to the next and previous node.

Whenever user input is required there should be some hint as to what is needed. This is the principle of using recognition rather than recollection when using user interfaces. Due to space constraints on the mobile device it is very difficult to give detailed instructions on the systems use. One way recognition is possible it that on every interface there is a menu, activated by the left soft-key that brings up a list of all commands accessible from that interface. These have clear names that allow the user to select the operation they require.

12.3.3 Control sequences

One important aspect of the user interfaces are the sequences of commands required to access the functionality provided in Daisy. The sequences of commands should be minimised in order to make it as easy as possible for the users to get what they want from the application. The intended users are unlikely to remember or investigate to find long sequences of commands to perform the operations they require. This aspect only concerns the user system as the administrators will be able to interpret the user interfaces much better and will be provided with user guides when the system is ready.

To analyse the control sequences used it is important to look at the number of actions required by the user to activate the required functions. Table 12.7 shows the functions available in the user system and the commands used to activate them. All control sequences begin at the main menu and are listed in appendix B.

Shortcuts minimise the length of control sequences. On each interface the main function for that interface has been associated with the select command.

The two longest control sequences are getting assistance and connecting a GPS device to use tracking. It is desired that these control sequences should be cut down. This may be possible through automation of the commands as described in section 12.3.4.

At any point in a control sequence it is possible for the user to return to
<table>
<thead>
<tr>
<th>Feature</th>
<th>Number of Control Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select first route</td>
<td>2</td>
</tr>
<tr>
<td>Select a new route</td>
<td>4</td>
</tr>
<tr>
<td>Show selected route as list</td>
<td>1</td>
</tr>
<tr>
<td>Show selected route in detail</td>
<td>2</td>
</tr>
<tr>
<td>Show selected route in detail and full screen</td>
<td>As above + 1</td>
</tr>
<tr>
<td>Show selected route with tracking</td>
<td>As above + 5</td>
</tr>
<tr>
<td>Request assistance from route details</td>
<td>4</td>
</tr>
<tr>
<td>Request assistance from other interfaces</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 12.7: Control sequences

the main screen by selecting the “Back” option associated with the right soft-key. This gives the user the advantage that if at any point they lose track of the control sequence they are following they can return to the main screen and start again.

12.3.4 Degree of automation

In order for the Daisy system to be more user friendly as much of the functionality should be automated as possible. There are already a number of ways in which this has been done, however there are many more ways in which it can be extended.

Existing Automation

Route loaded automatically: when the system is loaded it tries to load a route for the user. Using the "Route" setting the system attempts to load in its last known route. Using this method the user will be able to go straight to the route that they have most recently used.

Go directly to current location in route: when tracking is enabled the system goes directly to the point in the route where the user is positioned based on location information from the GPS. This saves the user have to scroll through the list of route nodes to find their current location.

Instruction media played automatically: as soon as a route node is displayed the instruction media (audio) associated with it will be played. This can then be repeated using commands from the interface.

Potential Automation

Location based route selection: to help the user select a route it may be a good idea to use a location obtained from the GPS device to find the routes that the user is either currently part way through or in close proximity to. This feature would be useful to the user as it is more likely that they would be interested in these routes given their current position and it would reduce the number of routes that they need to choose between making the entire process much easier for them.

107
Connect to the GPS device straight away: in order to reduce the number of operations required by the user to get the system into a state where the user can follow a route it may be better to start the tracker and connect to the GPS device automatically rather than waiting for the user to select the tracking command. This would assist the user as the system would immediately recognise their location and display the correct portion of the route.

Although it may not be possible to connect to the GPS device it would be useful to at least attempt the connection. If the connection were to fail it would be required that the system continued as it would have without the tracker allowing the user to manually navigate the route.

Assistance message sent automatically: this was described in section 12.1.4.

12.4 Evaluation Scenarios

Several evaluation scenarios were specified in the evaluation criteria chapter 4. A volunteer was used to test the system in the two different scenarios. The user was given minimal help in using the system after being given an explanation of how the system operates as a whole. The idea behind this was to discover whether or not they could infer how the system works without being given specific instructions.

Figure 12.2: Daisy being used to capture routes

12.4.1 Basic Navigation

The first scenario used a sample route programmed into the system that the user was asked to follow. The system operated in the second tracking mode developed as detailed in the user guide C. This meant that from a start point the user is given an instruction to get to the next route node, when they arrive there they are notified by the system that they are in the correct location. They then line themselves up using the orientation media before viewing the next node and repeating the process.
Although the test user was not from the set of intended Daisy users (those with cognitive difficulties that Daisy intends to overcome) the feedback they provided in these tests is still a useful evaluation of the system. As someone unfamiliar with the application and how it works they were able to give some idea of what it will be like for a Daisy user as it can be assumed that whenever they use the system it is in a way as if they were always unfamiliar with it.

User comments on evaluation scenario

Following the evaluation a discussion was held concerning the use of the system and any parts of the system that were found to have problems.

Main route display good: once you had received the instructions required the main screen, in full screen mode was very useful. It may be better to simply use full screen mode rather than having to move into this mode.

Tracking indicator: the tracking indicator using colours worked well. It was clear when you were moving in the correct direction (green edges to screen) and also when you the wrong direction (red edges). The notification that you had arrived at the correct point was also good as you then knew it was time to line up the orientation media.

Direction to be taken: when following a route there is no concept of the direction that should be taken after the initial instructions have been provided. Even though the system does indicate if you are going in the correct direction (green background) or wrong direction (red background) it would be more useful if there was an indication of how to correct yourself.

Distance from target: in order to let you know how far you are from your target location it would be useful to have a pictorial output that lets you know when you are almost at your location.

Easy to press the wrong button: at times it is easy to press the wrong button and cause the system to change the screen to something that you do not want.
Tracking problems: occasionally the GPS signal would be lost. When this happened there was no indication on the system that you were not being tracked. Then when the GPS signal returned, very often it reported the wrong location so it would say you have gone of track initially before receiving the correct location value and saying that you were on the right course.

Discussion of scenario performance

The user was able to follow the route specified by Daisy. Using only what they could see on the screen they were able to select the desired route and then show it so that it could be followed. Once activated tracking could be followed using the instructions provided by the system to reach the destination.

The main concerns raised in the evaluation were associated with the route node display. The following improvement to this display came from this evaluation.

Improved Route Node Display

This evaluation scenario has identified how the route node display can be improved. For a start the display should always be in full screen mode. As the menus do not work in full screen mode these should be removed and the interface should only take input from the keypad and joystick.

Rather than always displaying orientation media which is only required when the user reaches their destination an arrow should be displayed pointing in the direction that the user should be moving. As the user gets close to the desired route node this arrow should get smaller. Only when the user arrives at the route node will the orientation media be displayed. Once the orientation media has been lined up the user can inform the system by pressing “Select”. This then displays the next route node. The workflow of the new display mode has been given in figure 12.4.

12.4.2 Specifying New Routes

The second evaluation scenario was to specify a new route for Daisy. This involved capturing media for the route and describing the route nodes used to make up the route. The administrator system was used to capture the new route.

User comments on evaluation scenario

Again the user gave feedback after capturing the route. The major points raised were:

Simple to program route: the interfaces and commands provided to specify a route were found to be simple to follow. It was easy to find the operations required due to the well named commands in the menu. Messages to prompt what to do next were useful without getting in the way once you knew what you were doing. The media library interface proved very useful
when selecting media to be used in routes. It was possible to quickly select pre-captured media as well as being easy to capture new media.

**Camera slow:** after specifying that capture was required there was a substantial delay between pressing the button, capturing the image and then storing this image into the media library. During this time it was not known what is happening and more importantly at what stage the camera actually takes the picture. It was found that you could end up waiting some time unsure of whether the image had been taken or not.

**Crashed at one point:** when capturing media the system at one point crashed. This was because the camera was still processing when the user was specified other commands.

**Discussion of scenario performance**

It was noted that at times the administrator system appeared to take no action after a command was pressed. This is because some of the operations in the administrator system take a few seconds to process. As mentioned above some indication should be given to the user that the command has been activated and is being dealt with. This problem has already been described above in section 12.2.5.

Unfortunately there is no way of speeding up the camera. It would be useful again to indicate to the user what was happening during the delay.
12.5 Daisy on other platforms and devices

Even though this prototype application of Daisy was intended for the Nokia 6600 it is desired that Daisy should run on a number of different platforms. In theory, as long as a device has a java implementation with the CLDC 1.0 configuration and MIDP 2.0 profile it should run Daisy. In order for the full complement of features to be available the device will also need the Mobile Media, Bluetoot and Wireless Messaging optional APIs.

Unfortunately it has not been possible to obtain different devices that are suitable to run Daisy on. Daisy has only been tested on the Nokia 6600 and a Nokia 3230. There was no difference between how Daisy executed on these two devices as they are both Symbian series 60 phones with almost identical Java implementations. Daisy ran in the same way, except slightly faster on the 3230. There were no problems with interfaces as the screen size is the same as the 6600.

In order to get some idea of how the system runs on other devices the Daisy user system has been tested on different emulators. The Daisy user system was run on a series 40 and a series 90 emulator. Screen shots from these executions can be seen in figures 12.5 and 12.6 respectively.

![Screen shots of Daisy on Nokia 6600 and Nokia 3230](image)

Figure 12.5: Symbian Series 40 (Nokia 6620) Emulator screen shots

It can be seen that the main problems executing on other devices are in the way the interfaces are displayed. As the screen sizes vary a lot there are
problems getting all the required information onto the smaller screen or the opposite view shows the much larger screen size not being used effectively.

What is needed is to dynamically position elements on the screen to ensure that all the available space is used rather than using the same displays for all devices.

As well as the screen sizes the methods of input differ on the different devices. Commands on interfaces are associated with different buttons on different devices. This means that the most useful buttons are not necessarily displayed in the best positions of different types of device.

The difference in command position is demonstrated well on the series 90 emulator as shown in figure 12.6. Here there are much larger command buttons for the commands that would have previously been in a menu. The commands displayed on these buttons are not those that would necessarily have been chosen had the application been designed with this device in mind. For example, on the route node display, the next and previous buttons are already controlled by the directional controls so it would be better to use the command buttons to allow the instruction media to be played or to connect a GPS and allow tracking.

12.6 Successes and Failures

As have been shown in this evaluation Daisy has had a variety of successes and failures. The main successes identified include:

- Ability to specify routes using a variety of media which can then be displayed back to the user providing both pre-trip planning in-trip reassurance.
• A tracking system that allows the user to follow the route, displaying directions using media according to their locations.

• Mobile route capture software to allow a travel trainer to specify detailed routes for a Daisy user.

• Docking station allowing data to be transferred from a device where memory is at a premium to a desktop computer with much more memory available allowing large collections of routes and associated media to be built up.

The major limitations in Daisy have been identified to be:

• The user system is not as robust as it was desired to be.

• The persistent store access is slow causing some functions in the system suffer from delays.

• When delays occur due to background processing there is no indication that something is still happening.

• The desktop administrator systems function is limited to only sending, receiving and storing routes. This should be extended to also manipulate the route data, creating and modifying routes to be sent to a mobile device.

• Daisy is not suitable for devices other than Symbian series 60 phones. Before being used on different devices all interfaces would need to be adjusted.
Part IV

Conclusions and Further Work
Chapter 13

Conclusion

Having looked back on what Daisy has already achieved in the evaluation it is now time to look forwards to what Daisy could go on to be in the future. It is also important to look at Daisy from a lower level, thinking about the development process and how it could have been improved on.

13.1 Further Work

13.1.1 Improvements to Daisy

The evaluation identified a number of areas where Daisy has had shortcomings. These areas can be improved upon to make Daisy into a product suitable for deployment to users.

- Improving the automation of certain aspects of the system to make it more user friendly.
- Modify the workflow used in displaying information to the user when tracking. Respond more to environment events, only providing information when necessary and hiding it at other times.
- Providing more information when tracking such that the user has a better idea of their progress.
- Improving the robustness of the system to ensure that it will endure all kinds of treatment from users.
- Create Daisy interfaces for different types of devices to provide a more general solution rather than one specifically for Symbian Series 60 phones.
- Extend the functionality of the desktop administrator system to allow creation and manipulation of routes which can then be deployed to mobile devices.
13.1.2 Extensions to Daisy

A number of extensions were proposed in the specification. At that time it was unsure as to whether these extensions were feasible. Using the experience gained from Daisy it is now possible to better comment on the practicality of these extensions.

Include mapping with Daisy

As longitude and latitude values are recorded for each route node it would be a fairly simple task to plot their locations onto a map. The problem would be in how to specify the map. As memory available is limited only a very small map could be stored on the device. It would be possible to store a limited local map for each route in order to show how the different route nodes link together.

Mapping is still not seen to be a useful feature for Daisy. It is not suitable for the users as a typical Daisy user would not be able to understand the information presented to them in the map. They could not use it to determine their location or find their way.

More powerful routing

More powerful routing would allow Daisy to create routes dynamically from a large set of route nodes given a start and destination point. The problem seen with using improved routing is in specifying all the nodes to be used. To allow dynamically chosen routes every route node requires instructions detailing how to get to all surrounding route nodes. Each instruction must be carefully thought out and provided with sufficient media to accurately describe it.

Problems to be overcome before this routing can be used include:

- How to capture large amounts of media to be used for instructions at each route node.
- Ensuring that all instructions are specific enough. The instructions used in Daisy routes must be considered very carefully so it is hard to quickly produce large numbers of them.
- How to join adjacent route nodes so that the instructions follow on from each other in a suitable way.

Predictive tracking based on location, heading and speed

Currently only the longitude and latitude provided by the GPS device are used. To improve tracking the bearing and velocity at which the user is travelling could also be used. This would allow the tracker to more accurately determine where the user is at any given moment. It could also be used to handle signal fluctuations as given that a user is travelling from a particular point in a particular direction you can identify anomalous values where the location has drastically changed.
13.2 Further Uses for the Daisy System

The application framework developed for Daisy is not limited to use by people with disabilities. Many people find it difficult to navigate when presented with new and different surroundings. The Daisy system could be used as a navigational tool in a number of situations.

Tourist routes: When people arrive in a new city as a tourist they often wish to follow a standard route between the major attractions. Daisy could be used to plot a route between the major attractions in a city. This would make a kind of electronic guide book helping the tourist to get their bearings.

In a similar way the Daisy system could be used to guide tourists from any given attraction to any other attraction. By dynamically downloading routes, for example from an information point via Bluetooth they could be given directions to wherever they wish to go.

Daisy would be suitable in this situation as in addition to giving directions the system could highlight points of interest along the way that tourists may otherwise have missed. Audio and image media could be used to describe landmarks of interest and highlight features of particular interest.

Directions to business premises: Very often people away on business require directions from local transportation to a particular company’s premises. Instead of downloading directions or a map from a company’s website you could download Daisy directions. These can then easily be followed to arrive at a defined location. The instructions would be in a clear, step by step format allowing the users to easily find a route to their destination.

13.3 Mobile Development

13.3.1 Choice of development language

If the project were repeated J2ME would still be used for development rather than the native C++. Despite a number of limitations faced during development the advantages are still considerable.

A set of tools such as J2ME Polish would be used to complement the J2ME development. This benefits development making it easier to perform certain functions such as:

- Pre-processing to make device specific implementations
- Optimisation of code
- Simple debugging and logging
- Rapid user interface development
13.3.2 Games Programming

At a recent Games Developer Event at Imperial College a talk was given titled “Mobile Gaming: What is it all about?”. The talk was given by Chris White from MacroSpace, a leading games developer for mobile phones. Following the presentation several questions were put to Chris White about the challenges faced in mobile gaming, specifically aimed at how challenges faced in this project. It was interesting to see the methods they use and problems faced are similar to those in Daisy.

Floating point numbers

Most games titles are developed using CLDC 1.0 meaning floating point number support is not available. Fixed Point arithmetic is used to make up for the lack of floating point precision. The benefit of this is that it is typically faster than floating point numbers anyway and the games titles being produced generally rely on fast execution. When more complex titles involving 3D graphics are developed for high end handsets, floating point support is used where available.

Variations between devices

Preprocessing is used by many developers in games development however MacroSpace choose an alternative approach. They consider use of device specific wrapper classes for particular handsets to be a much cleaner approach to this problem. This means that their code-base consists of the core shared classes as well as a few device specific ones.

The device specific classes contain some final static values specific to individual handsets. These constant values can be used in computations to ensure that all handsets behave in the same way.

Porting is considered the most difficult aspect with regard to mobile development. Targeting a game for one device is simple but then ensuring that the same game works correctly on up to 50 handsets is a lot more difficult. In mobile games development they have dedicated teams whose job it is to adapt initial versions of games such that they will work on multiple platforms and devices.

13.4 Final Thoughts

The original specification gave the following description of Daisy:

“A piece of software will be developed to give users pre-trip planning and in-trip reassurance enabling them to successfully navigate the public realm.”

This has been provided. It is possible to use Daisy to both study different routes and then use a GPS tracking system to follow the route to navigate the public realm. Even though there are a number of issues regarding how the routes are stored and presented a basis has been provided to show what is possible. Should the improvements suggested above be implemented it is thought that Daisy will be ready for use with the intended users.
It is hoped that Daisy will be used in the future by its intended users. If not directly, the ideas learnt in this project should be taken away and used in development future assistive technology projects. The work put into this project is easily justified when you consider the freedom and independence it could bring to peoples live.
Part V

Appendix
Appendix A

Tracking around Queens Tennis Club

This appendix shows the results of tracking a route around Queens Tennis club.

The values were taken from the “GPS Status” interface which shows the current values obtained by the tracker. Once the points had been recorded they were plotted onto a map of the area.

The route followed is shown in figure A.1. Along the route nine location values were recorded. These are given in table A.1. The precise location of the recorded locations have been plotted in figure A.2. It can be clearly seen that the locations of the sample points follow the route, the points even appear accurate to which side of the road the reading was taken from.

Figure A.1: Route followed around Queens tennis club
<table>
<thead>
<tr>
<th>Point</th>
<th>Desc</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Entrance to Hammersmith Cemetery</td>
<td>51.48805 North, -0.21399 West</td>
</tr>
<tr>
<td>2</td>
<td>End of Field Road</td>
<td>51.48658 North, -0.21331 West</td>
</tr>
<tr>
<td>3</td>
<td>Rear entrance to Queens Club</td>
<td>51.48644 North, -0.21113 West</td>
</tr>
<tr>
<td>4</td>
<td>St Andrews Road</td>
<td>51.48710 North, -0.20914 West</td>
</tr>
<tr>
<td>5</td>
<td>Side Entrance to Queens Club</td>
<td>51.48741 North, -0.20924 West</td>
</tr>
<tr>
<td>6</td>
<td>Comeragh Road</td>
<td>51.48891 North, -0.21078 West</td>
</tr>
<tr>
<td>7</td>
<td>Charleville Road</td>
<td>51.48833 North, -0.20906 West</td>
</tr>
<tr>
<td>8</td>
<td>Perham Road</td>
<td>51.48770 North, -0.20818 West</td>
</tr>
<tr>
<td>9</td>
<td>Normand Road</td>
<td>51.48679 North, -0.20852 West</td>
</tr>
</tbody>
</table>

Table A.1: Locations on path around Queens Club

Figure A.2: Sample location points collected on the route
Appendix B

Control Sequences

The control sequences to activate different features include:

• Select first route
  – Press “Select”
  – Scroll down list to desired item
  – Press “Select”

• Select a new route
  – Open menu (left soft-key)
  – Scroll down 1 option to “Select Route” command
  – Press “Select”
  – Scroll down list to desired route
  – Press “Select”

• Show selected route as list
  – Press “Select”

• Show selected route in detail
  – Press “Select”
  – Scroll down list then to desired route node
  – Press “Select”

• Show selected route in detail and full screen
  – As above to show
  – Press “Up” to go to full screen mode

• Show selected route with tracking
  – As above to show route node
  – Open menu (left soft-key)
– Scroll down 3 options to “Connect GPS” command
– Press “Select”

● Request assistance from route details
  – Press “0”
  – Open menu (left soft-key)
  – Scroll down 1 option to “Yes” command
  – Press “Select”

● Request assistance from other interfaces
  – Open menu (left soft-key)
  – Scroll down 2 options to “Assistance” command
  – Press “Select”
  – Scroll down 1 option to “Yes” command
  – Press “Select”
Appendix C

User Guide

In order to get an idea about how the Daisy system works this user guide has been provided. The user guide describes sample usage of the Daisy User System on the Nokia 6600.

C.1 Basic Controls

The joystick has four directions: up, down, left and right. The select action is when you press down in the middle of the joystick.

Selecting a command from the menu is performed by pressing the left soft-key to bring up the menu and then using the up and down direction controls on the joystick to move through the list. A command is selected from the list by pressing down on the joystick.

C.2 Starting the Application

To start the application select the “Daisy user system” icon from the main phone menu. This brings up the Daisy user system main screen as shown in figure C.1.

![Figure C.1: Daisy user main screen](image)
C.3 Selecting a Route

To select a route open the route selection interface. This is done using the “Select Route” command in the left soft-key menu.

The route selection list displays all possible Daisy routes that can be followed, see figure C.2.

To select a route from the list scroll up and down the list using the joystick until the route that you desire to use is highlighted. When highlighted press down on the joystick to select the route.

C.4 Display the route

The first route display is a list of all the nodes in the route. This is shown in figure C.3. This interface is shown after a route is selected, by pressing select from the main Daisy screen or by selecting the “Show Route” command from the main user screen.

To view a particular node in more detail scroll through the list until it is highlighted and then press the select button again. This action brings up the single route node display, see figure C.4. When the route node is displayed the instruction media is played. As well as viewing this screen with the title bar it is possible to view it in full screen mode by pressing “Up”.

There are a number of commands that can be used to control the route node display.
Figure C.4: Daisy route node display (both in with title bar and fullscreen)

**SELECT**: this plays the instruction media.

**UP**: this moves in and out of full screen mode.

**DOWN**: this displays the description for the route node as in figure C.5.

**LEFT**: go to the previous route node.

**RIGHT**: go to the next route node.

**0**: send an assistance message.

Figure C.5: Description display

All of these commands are accessible through the shortcuts given above and the menu. When in full screen mode the menu does not work.

### C.5 Tracking

Tracking can be enabled using a GPS. The tracking mode is activated by selecting “Connect GPS” from the menu. Locations are then obtained from the GPS device and reported to Daisy.

When tracking the background of the route node interface will be coloured in to indicate position.

- A solid blue background indicates you have arrived at the route node.
• Green bars at the top and bottom indicate that you are moving in the correct direction.

• red bars at the top and bottom indicate that you are moving in the wrong direction.

C.6 Assistance

The assistance interface can be brought up from any point in the user system by selecting the assistance option from the menu. This goes to the assistance interface shown in figure C.6.

![Assistance Interface](image)

Pressing select or selecting “Yes” from the menu sends an assistance menu. Pressing the right soft-key goes back to the previous interface without sending an assistance message.
Bibliography


